Docket No. 251092US2SRD

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Tatsunori KANAI, et al. GAU:

SERIAL NO: NEW APPLICATION EXAMINER:

FILED: HEREWITH

FOR: INFORMATION PROCESSING SYSTEM INCLUDING PROCESSORS AND MEMORY MANAGING METHOD USED IN THE SAME SYSTEM

REQUEST FOR PRIORITY

COMMISSIONER FOR PATENTS
ALEXANDRIA, VIRGINIA 22313

SIR:

☐ Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §120**.

☐ Full benefit of the filing date(s) of U.S. Provisional Application(s) is claimed pursuant to the provisions of **35 U.S.C. §119(e)**: **Application No.** **Date Filed**

■ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of **35 U.S.C. §119**, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY	APPLICATION NUMBER	MONTH/DAY/YEAR
Japan	2003-185416	June 27, 2003

Certified copies of the corresponding Convention Application(s)

■ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

☐ were filed in prior application Serial No. filed

☐ were submitted to the International Bureau in PCT Application Number
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.

☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and

☐ (B) Application Serial No.(s)

☐ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Marvin J. Spivak
Registration No. 24,913

C. Irvin McClelland
Registration Number 21,124

Customer Number
22850
Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 05/03)

03S1440

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　６月２７日

出願番号
Application Number: 特願２００３－１８５４１６

[ST. 10/C]: [ＪＰ２００３－１８５４１６]

出願人
Applicant(s): 株式会社東芝

２００３年１０月　７日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００３－３０８２５９９

【書類名】　特許願
【整理番号】　A000303289
【提出日】　平成15年 6月27日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F 15/00
【発明の名称】　情報処理システムおよびメモリ管理方法
【請求項の数】　12
【発明者】
【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
【氏名】　金井　達徳
【発明者】
【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
【氏名】　前田　誠司
【発明者】
【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
【氏名】　吉井　謙一郎
【特許出願人】
【識別番号】　000003078
【氏名又は名称】　株式会社　東芝
【代理人】
【識別番号】　100058479
【弁理士】
【氏名又は名称】　鈴江　武彦
【電話番号】　03-3502-3181

【選任した代理人】
　【識別番号】　100091351
　【弁理士】
　【氏名又は名称】　河野　哲
【選任した代理人】
　【識別番号】　100088683
　【弁理士】
　【氏名又は名称】　中村　誠
【選任した代理人】
　【識別番号】　100108855
　【弁理士】
　【氏名又は名称】　蔵田　昌俊
【選任した代理人】
　【識別番号】　100084618
　【弁理士】
　【氏名又は名称】　村松　貞男
【選任した代理人】
　【識別番号】　100092196
　【弁理士】
　【氏名又は名称】　橋本　良郎
【手数料の表示】
　【予納台帳番号】　011567
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　情報処理システムおよびメモリ管理方法

【特許請求の範囲】

【請求項１】　第１のローカルメモリを有する第１のプロセッサと、

第２のローカルメモリを有する第２のプロセッサと、

第３のローカルメモリを有する第３のプロセッサと、

前記第１のプロセッサによって実行される第１のスレッドの実効アドレス空間の一部に、前記第１のスレッドとの相互作用を行う第２のスレッドが実行される前記第２および第３の一方のプロセッサに対応する前記第２および第３の一方のローカルメモリをマッピングする手段と、

前記第２のスレッドが実行されるプロセッサが前記第２および第３の一方のプロセッサから他方のプロセッサに変更された場合、前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第２および第３の一方のローカルメモリから他方のローカルメモリに変更する手段とを具備することを特徴とする情報処理システム。

【請求項２】　前記第１のプロセッサ、前記第２のプロセッサ、および前記第３のプロセッサによって共有される共有メモリと、

前記第２のスレッドの実行が停止される場合、前記第２のスレッドが実行されていた前記第２および第３の一方のプロセッサに対応する前記第２および第３の一方のローカルメモリの内容を前記共有メモリ上のメモリ領域に保存する手段と、

前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第２および第３の一方のローカルメモリから前記共有メモリ上のメモリ領域に変更する手段とをさらに具備することを特徴とする請求項１記載の情報処理システム。

【請求項３】　前記第２のスレッドが前記第２および第３の一方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第２および第３の一方のプロセッサに対応する前記第２および第３の一方のローカルメモリに復元する手段と、

前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第２および第３の一方のローカルメモリに変更する手段とをさらに具備することを特徴とする請求項２記載の情報処理システム。

【請求項４】　前記第２のスレッドが前記第２および第３の他方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第２および第３の他方のプロセッサに対応する前記第２および第３の他方のローカルメモリに復元する手段と、

前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第２および第３の他方のローカルメモリに変更する手段とをさらに具備することを特徴とする請求項２記載の情報処理システム。

【請求項５】　第１のローカルメモリを有する第１のプロセッサ、第２のローカルメモリを有する第２のプロセッサ、および第３のローカルメモリを有する第３のプロセッサを含む情報処理システムにおいてスレッド間の通信に用いられるローカルメモリを管理するメモリ管理方法であって、

前記第１のプロセッサによって実行される第１のスレッドの実効アドレス空間の一部に、前記第１のスレッドとの相互作用を行う第２のスレッドが実行される前記第２および第３の一方のプロセッサに対応する前記第２および第３の一方のローカルメモリをマッピングするステップと、

前記第２のスレッドが実行されるプロセッサが前記第２および第３の一方のプロセッサから他方のプロセッサに変更された場合、前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第２および第３の一方のローカルメモリから他方のローカルメモリに変更するステップとを具備することを特徴とするメモリ管理方法。

【請求項６】　前記第２のスレッドの実行が停止される場合、前記第２のスレッドが実行されていた前記第２および第３の一方のプロセッサに対応する前記第２および第３の一方のローカルメモリの内容を、前記第１のプロセッサ、前記第２のプロセッサ、および前記第３のプロセッサによって共有される共有メモリ

上のメモリ領域に保存するステップと、

前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第２および第３の一方のローカルメモリから前記共有メモリ上のメモリ領域に変更するステップとをさらに具備することを特徴とする請求項５記載のメモリ管理方法。

【請求項７】　前記第２のスレッドが前記第２および第３の一方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第２および第３の一方のプロセッサに対応する前記第２および第３の一方のローカルメモリに復元するステップと、

前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第２および第３の一方のローカルメモリに変更するステップとをさらに具備することを特徴とする請求項６記載のメモリ管理方法。

【請求項８】　前記第２のスレッドが前記第２および第３の他方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第２および第３の他方のプロセッサに対応する前記第２および第３の他方のローカルメモリに復元するステップと、

前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第２および第３の他方のローカルメモリに変更するステップとをさらに具備することを特徴とする請求項６記載のメモリ管理方法。

【請求項９】　第１のローカルメモリを有する第１のプロセッサ、第２のローカルメモリを有する第２のプロセッサ、および第３のローカルメモリを有する第３のプロセッサを含むコンピュータに、スレッド間の通信に用いられるローカルメモリを管理させるプログラムであって、

前記第１のプロセッサによって実行される第１のスレッドの実効アドレス空間の一部に、前記第１のスレッドとの相互作用を行う第２のスレッドが実行される前記第２および第３の一方のプロセッサに対応する前記第２および第３の一方のローカルメモリをマッピングする処理を、前記コンピュータに実行させる手順と

前記第２のスレッドが実行されるプロセッサが前記第２および第３の一方のプロセッサから他方のプロセッサに変更された場合、前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第２および第３の一方のローカルメモリから他方のローカルメモリに変更する処理を、前記コンピュータに実行させる手順とを具備することを特徴とするプログラム。

【請求項１０】　前記第２のスレッドの実行が停止される場合、前記第２のスレッドが実行されていた前記第２および第３の一方のプロセッサに対応する前記第２および第３の一方のローカルメモリの内容を、前記第１のプロセッサ、前記第２のプロセッサ、および前記第３のプロセッサによって共有される共有メモリ上のメモリ領域に保存する処理を、前記コンピュータに実行させる手順と、

前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第２および第３の一方のローカルメモリから前記共有メモリ上のメモリ領域に変更する処理を、前記コンピュータに実行させる手順とをさらに具備することを特徴とする請求項９記載のプログラム。

【請求項１１】　前記第２のスレッドが前記第２および第３の一方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第２および第３の一方のプロセッサに対応する前記第２および第３の一方のローカルメモリに復元する処理を、前記コンピュータに実行させる手順と、

前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第２および第３の一方のローカルメモリに変更する処理を、前記コンピュータに実行させる手順とをさらに具備することを特徴とする請求項１０記載のプログラム。

【請求項１２】　前記第２のスレッドが前記第２および第３の他方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第２および第３の他方のプロセッサに対応する前記第２および第３の他方のローカルメモリに復元する処理を、前記コンピュータに実行させる手順と、

前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第２および第３の他方のロー

カルメモリに変更する処理を、前記コンピュータに実行させる手順とをさらに具備することを特徴とする請求項１０記載のプログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は複数のプロセッサを含む情報処理システムおよび同システムで用いられるメモリ管理方法に関する。

【０００２】

【従来の技術】

従来より、サーバコンピュータのような計算機システムにおいては、その演算処理能力の向上を図るために、マルチプロセッサ、並列プロセッサのようなシステムアーキテクチャが利用されている。マルチプロセッサおよび並列プロセッサのどちらも、複数のプロセッサユニットを利用することによって演算処理の並列化を実現している。

複数のプロセッサユニットを備えたシステムとしては、例えば、１台の高速ＣＰＵ、複数台の低速ＣＰＵ、および共有メモリを備えたシステムが知られている（例えば、特許文献１参照）。このシステムにおいては、高速ＣＰＵおよび複数台の低速ＣＰＵに対する処理プログラムのプロセス群の割付は、プロセス群の並列動作度の大小および処理時間の大小に応じて行われる。

【０００３】

ところで、最近では、計算機システムのみならず、例えば、ＡＶ（オーディオ・ビデオ）データのような大容量のデータをリアルタイムに処理する組み込み機器においても、その演算処理能力の向上のためにマルチプロセッサ、並列プロセッサのようなシステムアーキテクチャの導入が要求されている。

【０００４】

【特許文献１】

特開平１０－１４３３８０号公報

【０００５】

【発明が解決しようとする課題】

しかし、複数のプロセッサを含むシステムアーキテクチャを前提としたリアルタイム処理システムの報告はほとんどなされていないのが現状である。

リアルタイム処理システムにおいては、ある許容時間時間の制限内に個々の処理を完了することが要求される。しかし、マルチプロセッサ、並列プロセッサのようなシステムアーキテクチャをリアルタイム処理システムに適用した場合においては、共有メモリに対するアクセスの競合、メモリバスのバンド幅の制約などにより、複数のプロセッサそれぞれの性能を十分に活用できなくなるという問題が生じる。また互いに異なるプロセッサによって実行されるスレッド間でデータを受け渡すための通信も共有メモリ上のバッファを介して行われるので、頻繁に相互作用を行うスレッド間においてはその通信に関するレイテンシが大きな問題となる。

【０００６】

本発明は上述の事情を考慮してなされたものであり、複数のプロセッサを用いて複数のスレッドを効率よく並列に実行することが可能な情報処理システムおよびメモリ管理方法を提供することを目的とする。

【０００７】

【課題を解決するための手段】

上述の課題を解決するため、本発明の情報処理システムは、第１のローカルメモリを有する第１のプロセッサと、第２のローカルメモリを有する第２のプロセッサと、第３のローカルメモリを有する第３のプロセッサと、前記第１のプロセッサによって実行される第１のスレッドの実効アドレス空間の一部に、前記第１のスレッドとの相互作用を行う第２のスレッドが実行される前記第２および第３の一方のプロセッサに対応する前記第２および第３の一方のローカルメモリをマッピングする手段と、前記第２のスレッドが実行されるプロセッサが前記第２および第３の一方のプロセッサから他方のプロセッサに変更された場合、前記第１のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第２および第３の一方のローカルメモリから他方のローカルメモリに変更する手段とを具備することを特徴とする。

【０００８】

この情報処理システムにおいては、各プロセッサにローカルメモリが設けられているので、各スレッドは共有メモリをアクセスせずとも、プロセッサ内のローカルメモリをアクセスするだけでプログラムを実行することができる。また、相互作用を行う相手のスレッドが実行されるプロセッサに応じて、実効アドレス空間にマッピングされる、相手のスレッドに対応するプロセッサのローカルメモリが自動的に変更されるので、各スレッドは相手のスレッドがディスパッチされるプロセッサを意識することなく、相手のスレッドとの相互作用を効率よく行うことが出来る。よって、複数のスレッドを効率よく並列に実行することが可能となる。

【０００９】

【発明の実施の形態】

以下、図面を参照して本発明の実施形態を説明する。

図１には、本発明の一実施形態に係るリアルタイム処理システムを実現するための計算機システムの構成例が示されている。この計算機システムは、リアルタイム性が要求される各種処理をその時間的な制約条件の範囲内で実行する情報処理システムであり、汎用計算機として利用できるほか、リアルタイム性が要求される処理を実行するための様々な電子機器用の埋め込みシステムとして利用することができる。図１に示されているように、この計算機システムにおいては、マスタープロセッサユニット（ＭＰＵ１１：Master Processing Unit）１１と、複数のバーサタイルプロセッサユニット（ＶＰＵ：Versatile Processing Unit）１２と、メインメモリ１４と、入出力制御装置１５とが、接続装置１３によって相互に接続されている。接続装置１３は、例えば、クロスバスイッチのような相互結合網、あるいはバスによって構成されている。リング状のバス構造を用いることも出来る。ＭＰＵ１１は計算機システムの動作を制御するメインプロセッサである。オペレーティングシステム(ＯＳ：Operating System)は、主にＭＰＵ１１によって実行される。ＯＳの一部の機能はＶＰＵ１２や入出力制御装置１５で分担して実行することもできる。各ＶＰＵ１２は、ＭＰＵ１１の管理の下で各種の処理を実行するプロセッサである。ＭＰＵ１１は、複数のＶＰＵ１２に処理を振り分けて並列に実行させるための制御を行う。これにより高速で効率よい処理

の実行を行うことが出来る。メインメモリ１４は、ＭＰＵ１１、複数のＶＰＵ１２および入出力制御装置１５によって共有される記憶装置（共有メモリ）である。ＯＳおよびアプリケーションプログラムはメインメモリ１４に格納される。入出力制御装置１５には、ひとつあるいは複数の入出力デバイス（入出力装置）１６が接続される。入出力制御装置１５はブリッジとも呼ばれる。

【００１０】

接続装置１５はデータ転送レートを保証するQoS機能を持つ。この機能は、接続装置１５を介したデータ転送を予約されたバンド幅（転送速度）で実行することによって実現される。QoS機能は、たとえば、あるＶＰＵ１２からメモリ１４に5Mbpsでライトデータを送信する場合、あるいはあるＶＰＵ１２と別のＶＰＵ１２との間で１００Mbpsでデータ転送する場合に利用される。ＶＰＵ１２は接続装置１３に対してバンド幅（転送速度）を指定（予約）する。接続装置１３は指定されたバンド幅を要求したＶＰＵ１２に対して優先的に割り当てる。あるＶＰＵ１２のデータ転送に対してバンド幅が予約されたならば、そのＶＰＵ１２によるデータ転送中に他のＶＰＵ１２、ＭＰＵ１１あるいは入出力制御装置１５が大量のデータ転送を行っても、予約されたバンド幅は確保される。この機能は、特に、リアルタイム処理を行う計算機にとって重要な機能である。

【００１１】

図１の構成では、ＭＰＵ１１が１つ、ＶＰＵ１２が４つ、メモリ１４が１つ、入出力制御装置が１つであるが、ＶＰＵ１２の個数は制限されない。またＭＰＵ１１を持たない構成も可能である。この場合、ＭＰＵ１１の行う処理は、ある一つのＶＰＵ１２が担当する。つまり、仮想的なＭＰＵ１１の役割をＶＰＵが兼ねる。

【００１２】

図２には、ＭＰＵ１１と各ＶＰＵ１２の構成が示されている。ＭＰＵ１１は処理ユニット２１およびメモリ管理ユニット２２を備えている。処理ユニット２１は、メモリ管理ユニット２２を通してメモリ１４をアクセスする。メモリ管理ユニット２２は、仮想記憶管理と、メモリ管理ユニット２２内のキャッシュメモリの管理を行うユニットである。各ＶＰＵ１２は、処理ユニット３１、ローカルス

トレージ（ローカルメモリ）３２、およびメモリコントローラ３３を備えている。各ＶＰＵ１２の処理ユニット３１は、そのＶＰＵ内部のローカルストレージ３２を直接アクセスすることができる。メモリコントローラ３３は、ローカルストレージ３２とメモリ１４の間のデータ転送を行うＤＭＡコントローラの役割を持つ。このメモリコントローラ３３は、接続装置１４のＱｏＳ機能を利用できるように構成されており、バンド幅を予約する機能および予約したバンド幅でデータ入出力を行う機能を有している。またメモリコントローラ３３は、ＭＰＵ１１のメモリ管理ユニット２２と同様の仮想記憶管理機能を持つ。ＶＰＵ１２の処理ユニット３１はローカルストレージ３２を主記憶として使用する。処理ユニット３１はメモリ１４に対して直接的にアクセスするのではなく、メモリコントローラ３３に指示して、メモリ１４の内容をローカルストレージ３２に転送して読んだり、ローカルストレージ３２の内容をメモリ１４に書いたりする。

【００１３】

ＭＰＵ１１のメモリ管理ユニット２２およびＶＰＵ１２のメモリコントローラ３３それぞれによって実行される仮想記憶管理は、たとえば図３のように実施することができる。ＭＰＵ１１の処理ユニット２１あるいはＶＰＵ１２のメモリコントローラ３３から見たアドレスは、図３の上の部分に示すような６４ビットのアドレスである。この６４ビットのアドレスは、上位の３６ビットがセグメント番号、中央の１６ビットがページ番号、下位の１２ビットがページオフセットである。このアドレスから、実際に接続装置１３を通してアクセスする実アドレス空間への変換は、セグメントテーブル５０およびページテーブル６０を用いて実行される。セグメントテーブル５０およびページテーブル６０は、メモリ管理ユニット２２およびメモリコントローラ３３に各々設けられている。

【００１４】

ＭＰＵ１１および各ＶＰＵ１２から見た実アドレス（ＲＡ）空間には、図４に示すように、たとえば以下のようなデータがマッピングされている。

１．メモリ（主記憶装置）

２．ＭＰＵ１１の各種制御レジスタ

３．各ＶＰＵ１２の各種制御レジスタ

４．各ＶＰＵ１２のローカルストレージ

５．各種入出力デバイス（入出力装置）の制御レジスタ（入出力制御装置の制御レジスタも含む）

ＭＰＵ１１および各ＶＰＵ１２は、実アドレス空間の該当するアドレスにアクセスすることで、１～５の各データを読み書きすることができる。特に、実アドレス空間にアクセスすることで、どのＭＰＵ１１からでも、あるいはどのＶＰＵ１２からでも、さらに入出力制御装置１５からでも、任意のＶＰＵ１２のローカルストレージ３２にアクセスすることができることは重要である。またセグメントテーブルあるいはページテーブルを用いて、ＶＰＵ１２のローカルストレージ３２の内容が自由に読み書きされないように保護することもできる。

ＭＰＵ１１あるいはＶＰＵ１２からみたアドレス空間は、図３の仮想記憶メカニズムを用いて、たとえば図５に示すようにマッピングされる。ＭＰＵ１１あるいはＶＰＵ１２上で実行しているプログラムから直接見えるのは、実効アドレス（ＥＡ；Effective Address）空間である。ＥＡは、セグメントテーブル５０によって、仮想アドレス（ＶＡ；Virtual Address）空間にマッピングされる。さらにＶＡは、ページテーブル６０によって、実アドレス（ＲＡ；Real Address）空間にマップされる。このＲＡが、図４で説明したような構造を持っている。

【００１５】

ＭＰＵ１１は制御レジスタ等のハードウェア機構によって、例えば、各ＶＰＵ１２のレジスタの読み書き、各ＶＰＵ１２のプログラムの実行開始／停止などの、各ＶＰＵ１２の管理を行うことができる。また、ＭＰＵ１１とＶＰＵ１２の間、あるいはあるＶＰＵ１２と他のＶＰＵ１２の間の通信や同期は、メールボックスやイベントフラグなどのハードウェア機構によって行うことが出来る。

【００１６】

この実施形態の計算機システムは、従来ハードウェアで実現されていたようなリアルタイム性の要求の厳しい機器の動作を、ソフトウェアを用いて実現することを可能にする。例えば、あるＶＰＵ１２があるハードウェアを構成するある幾つかのハードウェアコンポーネントに対応する演算処理を実行し、それと並行して、他のＶＰＵ１２が他の幾つかのハードウェアコンポーネントに対応する演算

処理を実行する。

【００１７】

図６はデジタルテレビ放送の受信機の簡略化したハードウェア構成を示している。図６においては、受信した放送信号はＤＥＭＵＸ（デマルチプレクサ）回路１０１によって音声データと映像データと字幕データそれぞれに対応する圧縮符号化されたデータストリームに分解される。圧縮符号化された音声データストリームはＡ－ＤＥＣ（音声デコーダ）回路１０２によってデコードされる。圧縮符号化された映像データストリームはＶ－ＤＥＣ（映像デコーダ）回路１０３によってデコードされる。デコードされた映像データストリームはＰＲＯＧ（プログレッシブ変換）回路１０５に送られ、そこでプログレッシブ映像信号に変換するためのプログレッシブ変換処理が施される。プログレッシブ変換された映像データストリームはＢＬＥＮＤ（画像合成）回路１０６に送られる。字幕データストリームはＴＥＸＴ（字幕処理）回路１０４によって字幕の映像に変換された後、ＢＬＥＮＤ回路１０６に送られる。ＢＬＥＮＤ回路１０６は、ＰＲＯＧ回路１０５から送られてくる映像と、ＴＥＸＴ回路１０４から送られてくる字幕映像とを合成して、映像ストリームとして出力する。この一連の処理が、映像のフレームレート（たとえば、１秒間に３０コマ、３２コマ、または６０コマ）に従って、繰り返し実行される。

【００１８】

図６のようなハードウェアの動作をソフトウェアによって実行するために、本実施形態では、たとえば図７に示すように、図６のハードウェアの動作をソフトウェアとして実現したプログラムモジュール１００を用意する。このプログラムモジュール１００は、複数の処理要素の組み合わせから構成されるリアルタイム処理を計算機システムに実行させるためのアプリケーションプログラムであり、マルチスレッドプログラミングを用いて記述されている。このプログラムモジュール１００は、図６のハードウェアコンポーネント群に対応する複数の処理要素それぞれに対応した手順を記述した複数のプログラム１１１～１１６を含んでいる。すなわち、プログラムモジュール１００には、ＤＥＭＵＸプログラム１１１、Ａ－ＤＥＣプログラム１１２、Ｖ－ＤＥＣプログラム１１３、ＴＥＸＴプログ

ラム１１４、ＰＲＯＧプログラム１１５、およびＢＬＥＮＤプログラム１１６が含まれている。ＤＥＭＵＸプログラム１１１、Ａ－ＤＥＣプログラム１１２、Ｖ－ＤＥＣプログラム１１３、ＴＥＸＴプログラム１１４、ＰＲＯＧプログラム１１５、およびＢＬＥＮＤプログラム１１６は、それぞれ図６のＤＥＭＵＸ回路１０１、Ａ－ＤＥＣ回路１０２、Ｖ－ＤＥＣ回路１０３、ＴＥＸＴ回路１０４、ＰＲＯＧ回路１０５、およびＢＬＥＮＤ回路１０６に対応する処理を実行するためのプログラムであり、それぞれスレッドとして実行される。つまり、プログラムモジュール１００の実行時には、ＤＥＭＵＸプログラム１１１、Ａ－ＤＥＣプログラム１１２、Ｖ－ＤＥＣプログラム１１３、ＴＥＸＴプログラム１１４、ＰＲＯＧプログラム１１５、およびＢＬＥＮＤプログラム１１６それぞれに対応するスレッドが生成され、生成されたスレッドそれぞれが１以上のＶＰＵ１２にディスパッチされて実行される。ＶＰＵ１２のローカルストレージ３２にはそのＶＰＵ１２にディスパッチされたスレッドに対応するプログラムがロードされ、スレッドはローカルストレージ３２上のプログラムを実行する。デジタルテレビ放送の受信機を構成するハードウェアモジュール群それぞれに対応するプログラム１１１～１１６と、構成記述１１７と呼ぶデータとをパッケージ化したものが、デジタルテレビ放送の受信機を実現するプログラムモジュール１００になる。

【００１９】

構成記述１１７は、プログラムモジュール１００内の各プログラム（スレッド）をどのように組み合わせて実行するべきかを示す情報であり、プログラム１１１～１１６間の入出力関係および各プログラムの処理に必要なコスト（時間）などを示す。図８には構成記述１１７の例が示されている。

【００２０】

図８の構成記述１１７の例では、スレッドとして動作する各モジュール（プログラムモジュール１００内の各プログラム）に対して、その入力につながるモジュール、その出力がつながるモジュール、そのモジュールの実行に要するコスト、出力がつながるモジュールそれぞれへの出力に必要なバッファサイズが記述されている。たとえば、番号③のＶ－ＤＥＣプログラムは、番号①のＤＥＭＵＸプログラムの出力を入力とし、その出力は番号⑤のＰＲＯＧプログラムに向かって

おり、その出力に必要なバッファは１ＭＢで、番号③のＶ－ＤＥＣプログラム自体の実行コストは５０であることを示している。なお、実行に必要なコストは、実行に必要な時間（実行期間）やステップ数などを単位として記述することができる。また、何らかの仮想的な仕様のＶＰＵで実行した場合の時間を単位とすることも可能である。計算機によってＶＰＵの仕様や処理性能が異なる場合もあるので、このように仮想的な単位を設けてコストを表現するのは望ましい形態である。図８に示した構成記述１１７に従って実行する場合の、プログラム間のデータの流れは図９の通りである。

【００２１】

さらに、構成記述１１７には、プログラム１１１～１１６それぞれに対応するスレッド間の結合属性を示す結合属性情報がスレッドパラメータとして記述されている。なお、スレッドパラメータはプログラム１１１～１１６中にコードとして直接記述することも可能である。

【００２２】

次に、図１０、図１１を参照して、プログラム１１１～１１６が本実施形態の計算機システムによってどのように実行されるかを説明する。ここでは、ＶＰＵ０とＶＰＵ１の２つのＶＰＵ１２が計算機システムに設けられている構成を想定する。毎秒３０フレームで映像を表示する場合の、各ＶＰＵ１２に対するプログラムの割り当てを時間を追って記入したのが図１０である。ここでは周期１の間で１フレーム分の音声と映像を出力している。まず、ＶＰＵ０でＤＥＭＵＸプログラムが処理を行い、その結果の音声と映像と字幕のデータをバッファに書き込む。その後ＶＰＵ１でＡ－ＤＥＣプログラムとＴＥＸＴプログラムを順次実行し、それぞれの処理結果をバッファに書き込む。ＶＰＵ０では、次にＶ－ＤＥＣプログラムが映像データの処理を行い、結果をバッファに書き込む。ＶＰＵ０では、続いてＰＲＯＧプログラムが処理を行い、結果をバッファに書き込む。この時点で、ＶＰＵ１でのＴＥＸＴの処理は終わっているので、最後のＢＬＥＮＤプログラムの実行をＶＰＵ０で行い、最終的な映像データを作成する。この処理の流れを、毎周期繰り返すように実行する。

【００２３】

ここで説明したように、所望の動作を滞りなく行えるように、各ＶＰＵ１２上で、いつ、どのプログラムを実行するかを決める作業を、スケジューリングとよぶ。スケジューリングを行うモジュールをスケジューラとよぶ。本実施形態では、プログラムモジュール１００中に含まれる上述の構成記述１１７に基づいてスケジューリングが行われる。

【００２４】

図１１は、毎秒６０フレームで表示する場合の実行の様子を示している。図１０と異なるのは、図１０では毎秒３０フレームだったので、１周期（１／３０秒）で１フレーム分の処理を完了できたのに対し、図１１では毎秒６０フレーム処理する必要がある点である。すなわち、１周期（１／６０秒）では１フレーム分の処理を完了できないので、図１１では、複数（ここでは２）周期にまたがったソフトウェアパイプライン処理を行っている。たとえば周期１のはじめに入力された信号に対して、ＶＰＵ０でＤＥＭＵＸ処理とＶ－ＤＥＣ処理を行う。その後、周期２においてＶＰＵ１でＡ－ＤＥＣ、ＴＥＸＴ、ＰＲＯＧ、ＢＬＥＮＤの各処理を行って最終的な映像データを出力する。周期２ではＶＰＵ０は次のフレームのＤＥＭＵＸとＶ－ＤＥＣの処理を行っている。このように、ＶＰＵ０によるＤＥＭＵＸ，Ｖ－ＤＥＣの処理と、ＶＰＵ１によるＡ－ＤＥＣ、ＴＥＸＴ、ＰＲＯＧ、ＢＬＥＮＤの処理を、２周期にまたがってパイプライン的に実行する。

【００２５】

なお、図７に示したプログラムモジュール１００は、本実施形態の計算機システムを組み込んだ機器内のフラッシュＲＯＭやハードディスクに予め記録しておいてもよいが、ネットワークを介して流通させるようにしてもよい。この場合、本実施形態の計算機システムによって実行される処理の内容は、ネットワークを介してダウンロードしたプログラムモジュールの種類に応じて決まる。よって、例えば本実施形態の計算機システムを組み込んだ機器に、様々な専用ハードウェアそれぞれに対応するリアルタイム処理を実行させることが出来る。例えば、新しいコンテンツの再生に必要な新しいプレーヤーソフトウェアやデコーダソフトウェアや暗号ソフトウェアなどを、本実施形態の計算機システムで実行可能なプログラムモジュールとして、コンテンツと一緒に配布することで、本実施形態の

計算機システムを搭載した機器であれば、いずれの機器でも、その能力が許す範囲内で、そのコンテンツを再生することができる。

【００２６】

（オペレーティングシステム）

本計算機システムでは、システム内にＯＳ（オペレーティングシステム）をひとつだけ実装する場合には、図１２に示すように、そのＯＳ２０１がすべての実資源（たとえば、ＭＰＵ１１、ＶＰＵ１２、メモリ１４、入出力制御装置１５、入出力装置１６など）を管理する。

一方、仮想計算機方式を用いて、複数のＯＳを同時に動作させることも可能である。この場合には、図１３に示すように、まず仮想計算機ＯＳ３０１を実装し、それがすべての実資源（たとえば、ＭＰＵ１１、ＶＰＵ１２、メモリ１４、入出力制御装置１５、入出力装置１６など）を管理する。仮想計算機ＯＳ３０１はホストＯＳと称されることもある。さらに仮想計算機ＯＳ３０１の上に、ひとつ以上のＯＳ（ゲストＯＳとも呼ぶ）を実装する。各ゲストＯＳ３０２，３０３は、図１４に示すように、仮想計算機ＯＳ３０１によって与えられる仮想的な計算機資源から構成される計算機上で動作し、ゲストＯＳ３０２，３０３の管理するアプリケーションプログラムに各種のサービスを提供する。図１４の例では、ゲストＯＳ３０２は、１つのＭＰＵ１１と、２つのＶＰＵ１２と、メモリ１４とから構成される計算機上で動いていると思っており、ゲストＯＳ３０３は１つのＭＰＵ１１と、４つのＶＰＵ１２と、メモリ１４とから構成される計算機上で動いていると思っている。ゲストＯＳ３０２からみたＶＰＵ１２や、ゲストＯＳ３０３からみたＶＰＵ１２が、実際には実資源のどのＶＰＵ１２に対応しているかは、仮想計算機ＯＳ３０１が管理している。ゲストＯＳ３０２，３０３は、その対応を意識する必要はない。

【００２７】

仮想計算機ＯＳ３０１は、計算機システム全体の資源を時分割で各ゲストＯＳ３０２，３０３に割り当てるように、ゲストＯＳ３０２，３０３のスケジューリングを行う。例えば、ゲストＯＳＯＳ３０２がリアルタイム処理を行うものであるとする。たとえば１秒間に３０回、正しいペースで処理を行いたい場合には、

各ゲストＯＳ３０２はそのパラメタを仮想計算機ＯＳ３０１に設定する。仮想計算機ＯＳ３０１は、１／３０秒に１回、確実にそのゲストＯＳ３０１に必要なだけの処理時間が割り当てられるようにスケジューリングを行う。リアルタイム性を要求しない処理を行うゲストＯＳには、リアルタイム性を要求するゲストＯＳよりも低い優先度で、処理時間の割り当てを行うように、スケジューリングが行われる。図１５は、時間軸を横にとって、ゲストＯＳ３０２とゲストＯＳ３０３が切り替わりながら動いている様子を示している。ゲストＯＳ３０２が動いている間は、ＭＰＵ１１と全てのＶＰＵ１２がゲストＯＳ３０２の資源として使用され、ゲストＯＳ３０３が動いている間は、ＭＰＵ１１と全てのＶＰＵ１２がゲストＯＳ３０３の資源として使用される。

【００２８】

図１６は別の動作モードを示している。ターゲットアプリケーションによってはＶＰＵ１２をずっと占有して利用したい場合がある。たとえば、常にデータやイベントを監視し続けることが必要なアプリケーションがこれに相当する。このようなときには、特定のＶＰＵ１２を特定のゲストＯＳによって占有するように、仮想計算機３０１のスケジューラがスケジュール管理する。図１６では、ＶＰＵ４をゲストＯＳ３０１の専用資源に指定した場合の例である。仮想計算機ＯＳ３０１がゲストＯＳ３０２（ＯＳ１）とゲストＯＳ３０３（ＯＳ２）を切り替えても、ＶＰＵ４は常にゲストＯＳ３０１（ＯＳ１）の管理下で動作し続ける。

【００２９】

さて、複数のＶＰＵ１２を用いてプログラムを動作させるために、本実施形態では、複数のＶＰＵ１２それぞれに割り当てるスレッドをスケジューリングするためのスケジューラを含む、ＶＰＵ実行環境と呼ぶソフトウェアモジュールを用いる。本計算機システムにＯＳがひとつしか搭載されていない場合は、図１７に示すようにそのＯＳ２０１にＶＰＵ実行環境４０１を実装する。この時、ＶＰＵ実行環境４０１は、ＯＳ２０１のカーネル内に実装することもできるし、ユーザプログラムレベルで実装することもできるし、両者に分割して協調して動作するように実装することも出来る。一方、仮想計算機ＯＳ上でひとつあるいは複数のＯＳを動作させる場合、ＶＰＵ実行環境４０１を実装する方式には、次のような

方式がある。

１．仮想計算機ＯＳ３０１の中にＶＰＵ実行環境４０１を実装する方式（図１８）

２．ＶＰＵ実行環境４０１を仮想計算機ＯＳ３０１が管理するひとつのＯＳとして実装する方式（図１９）。図１９では、仮想計算機ＯＳ３０１上で動作するゲストＯＳ３０４自体がＶＰＵ実行環境４０１である。

３．仮想計算機ＯＳ３０１が管理する各ゲストＯＳに、それぞれ専用のＶＰＵ実行環境４０１を実装する方式（図２０）。図２０においては、ゲストＯＳ３０２，３０３にそれぞれＶＰＵ実行環境４０１，４０２が実装されている。ＶＰＵ実行環境４０１，４０２は、仮想計算機ＯＳ３０１の提供するゲストＯＳ間の通信機能を用いて、必要に応じて、互いに連携して動作する。

４．仮想計算機ＯＳ３０１が管理するゲストＯＳのうちのひとつにＶＰＵ実行環境４０１を実装して、ＶＰＵ実行環境を持たないゲストＯＳは、仮想計算機ＯＳ３０１の提供するゲストＯＳ間の通信機能を用いて、ＶＰＵ実行環境４０１を持つゲストＯＳのＶＰＵ実行環境４０１を利用する方式（図２１）。

【００３０】

これらの方式のメリットは以下のとおりである。

方式１のメリット

・仮想計算機ＯＳの持つゲストＯＳ（仮想計算機ＯＳが管理する対象のＯＳ）のスケジューリングと、ＶＰＵ１２のスケジューリングを一体化できるので、効率良く、きめ細かなスケジューリングができ、資源を有効利用できる。

・複数のゲストＯＳ間でＶＰＵ実行環境を共有できるので、新しいゲストＯＳを導入する場合に新しくＶＰＵ実行環境を作らなくてもよい。

方式２のメリット

・仮想計算機ＯＳの上にあるゲストＯＳ間でＶＰＵ１２のスケジューラを共有できるので、効率良く、きめ細かなスケジューリングができ、資源を有効利用できる。

・複数のゲストＯＳ間でＶＰＵ実行環境を共有できるので、新しいゲストを導入する場合に新しくＶＰＵ実行環境を作らなくてもよい。

・ＶＰＵ実行環境を仮想計算機ＯＳや特定のゲストＯＳに依存せずに作れるので、標準化がしやすく、取り替えて使うことも出来る。特定の組み込み機器に適応したＶＰＵ実行環境を作って、その機器の特性を活かしたスケジューリング等を行うことで、効率良い実行ができる。

方式３のメリット

・各ゲストＯＳに対してＶＰＵ実行環境を最適に実装できるので、効率良く、きめ細かなスケジューリングができ、資源を有効利用できる。

【００３１】

方式４のメリット

・すべてのゲストＯＳがＶＰＵ実行環境を実装する必要がないので、新しいゲストＯＳを追加しやすい。

このように、いずれの方式でもＶＰＵ実行環境を実装することができる。また、このほかにも適宜実施可能である。

【００３２】

（サービスプロバイダ）

本実施形態の計算機システムにおいては、ＶＰＵ実行環境４０１は、各ＶＰＵ１２に関連する各種資源（各ＶＰＵの処理時間、メモリ、接続装置のバンド幅、など）の管理とスケジューリング機能の他に、さまざまなサービス（ネットワークを使った通信機能、ファイルの入出力機能、コーデックなどのライブラリ機能の呼び出し、ユーザとのインタフェース処理、入出力デバイスを使った入出力処理、日付や時間の読み出し、など）を提供する。これらのサービスは、ＶＰＵ１２上で動作するアプリケーションプログラムから呼び出されて、簡単なサービスの場合にはそのＶＰＵ１２上のサービスプログラムで処理される。しかし通信やファイルの処理などＶＰＵ１２だけでは処理できないサービスに関しては、ＭＰＵ１１上のサービスプログラムによって処理する。このようなサービスを提供するプログラムを、サービスプロバイダ（ＳＰ）と呼ぶ。

【００３３】

図２２にＶＰＵ実行環境のひとつの実施例を示す。ＶＰＵ実行環境の主要部分はＭＰＵ１１上に存在する。これが、ＭＰＵ側ＶＰＵ実行環境５０１である。各

ＶＰＵ１２上には、そのＶＰＵ１２内で処理可能なサービスを実行する最小限の機能のみを持つＶＰＵ側ＶＰＵ実行環境５０２が存在する。ＭＰＵ側ＶＰＵ実行環境５０１の機能は、大きく、ＶＰＵコントロール５１１と、サービスブローカ５１２の２つに分けられる。ＶＰＵコントロール５１２は、主に、各ＶＰＵ１２に関連する各種資源（ＶＰＵの処理時間、メモリ、仮想空間、接続装置のバンド幅、など）の管理機構や、同期機構や、セキュリティの管理機構や、スケジューリング機能を提供する。スケジューリング結果に基づいてＶＰＵ１２上のプログラムのディスパッチを行うのは、このＶＰＵコントロール５１１である。サービスブローカ５１２は、ＶＰＵ１２上のアプリケーションが呼び出したサービス要求を受けて、適当なサービスプログラム（サービスプロバイダ）を呼び出してそのサービスを提供する。

ＶＰＵ側ＶＰＵ実行環境５０２は、主に、ＶＰＵ１２上のアプリケーションプログラムが呼び出したサービス要求を受けて、ＶＰＵ１２内で処理できるものは処理し、そうでないものはＭＰＵ側ＶＰＵ実行環境５０１のサービスブローカ５１２に処理を依頼する働きをする。

【００３４】

図２３に、ＶＰＵ側ＶＰＵ実行環境５０２がサービス要求を処理する手順を示す。ＶＰＵ側ＶＰＵ実行環境５０２はアプリケーションプログラムからのサービス呼び出しを受け取ると（ステップＳ１０１）、ＶＰＵ実行環境５０２内で処理できるサービスであるかどうかを判別し（ステップＳ１０２）、それであれば、対応するサービスを実行して、結果を呼び出し元へ返す（ステップＳ１０３，Ｓ１０７）。一方、ＶＰＵ実行環境５０２内で処理できるサービスではないならば、該当するサービスを実行可能なサービスプログラムがＶＰＵ１２上で実行可能なプログラムとして登録されているかどうかを判断する（ステップＳ１０４）。登録されているならば、当該サービスプログラムを実行し、結果を呼び出し元へ返す（ステップＳ１０５，Ｓ１０７）。登録されていないならば、サービスブローカ５１２に処理を依頼し、そしてサービスブローカ５１２から返されるサービスの結果を呼び出し元へ返す（ステップＳ１０６，Ｓ１０７）。

【００３５】

図２４に、ＭＰＵ側ＶＰＵ実行環境５０１のサービスブローカ５１２が、ＶＰＵ側ＶＰＵ実行環境５０２から要求されたサービスを処理する手順を示す。サービスブローカ５１２はＶＰＵ側ＶＰＵ実行環境５０２からのサービス呼び出しを受け取ると（ステップＳ１１１）、ＶＰＵ実行環境５０１内で処理できるサービスであるかどうかを判別し（ステップＳ１１２）、それであれば、対応するサービスを実行して、結果を呼び出し元のＶＰＵ側ＶＰＵ実行環境５０２へ返す（ステップＳ１１３，Ｓ１１４）。一方、ＶＰＵ実行環境５０１内で処理できるサービスではないならば、該当するサービスを実行可能なサービスプログラムがＭＰＵ１１上で実行可能なプログラムとして登録されているかどうかを判断する（ステップＳ１１４）。登録されているならば、当該サービスプログラムを実行し、結果を呼び出し元のＶＰＵ側ＶＰＵ実行環境５０２へ返す（ステップＳ１１６，Ｓ１１４）。登録されていないならば、エラーを呼び出し元のＶＰＵ側ＶＰＵ実行環境５０２へ返す（ステップＳ１１７）。

【００３６】

なお、ＶＰＵ１２で実行するプログラムが発行するサービス要求には、サービスの実行結果のリプライを返すものもあれば、要求を出すだけでリプライの無いものもある。また、リプライ先は、通常は要求を出したスレッドであるが、リプライ先として他のスレッド、スレッドグループ、あるいはプロセスを指定することもできる。そのため、サービス要求のメッセージには、リプライ先の指定も含めることが好ましい。サービスブローカ５１２は、広く使われているオブジェクトリクエストブローカを用いて実現することができる。

【００３７】

（リアルタイム処理）

本実施形態の計算機システムはリアルタイム処理システムとして機能する。この場合、そのリアルタイム処理システムの対象とする処理は、大きく、

１．ハードリアルタイム処理

２．ソフトリアルタイム処理

３．ベストエフォート処理（ノンリアルタイム処理）

の３種類に分類できる。１と２がいわゆるリアルタイム処理と呼ばれるものであ

る。本実施形態のリアルタイム処理システムは、多くの既存のＯＳと同様、スレッドとプロセスの概念を持っている。ここではまず、本実施形態のリアルタイム処理システムにおけるスレッドとプロセスに関して説明する。

【００３８】

スレッドには、次の３つのクラスがある。

１．ハードリアルタイムクラス

このスレッドクラスは、その時間要件（timing requirements）は非常に重要で、その要件が満たされなかった際に重大な状況になるような、重要なアプリケーションに用いる。

２．ソフトリアルタイムクラス

このスレッドクラスは、例えその時間要件が満たされなかった場合においても、その品質が低下するだけのアプリケーションに用いる。

３．ベストエフォートクラス

このスレッドクラスは、その要件の中に一切の時間要件を含まないアプリケーションに用いる。

【００３９】

スレッドは本アルタイム処理システム内において処理を実行する実体である。スレッドには、そのスレッドが実行するプログラムが関連付けられている。各スレッドは、スレッドコンテクストと呼ぶ、それぞれのスレッドに固有の情報を保持している。スレッドコンテクストには、たとえば、プロセッサのレジスタの値や、スタックなどの情報が含まれている。

本リアルタイム処理システムにおいては、ＭＰＵスレッドとＶＰＵスレッドの２種類のスレッドが存在する。これら２つのスレッドは、そのスレッドが実行されるプロセッサ（ＭＰＵ１１かＶＰＵ１２）によって分類されており、スレッドとしてのモデルは全く同じである。ＶＰＵスレッドのスレッドコンテクストには、ＶＰＵ１２のローカルストレージ３２の内容や、メモリコントローラ３３の持つＤＭＡコントローラの状態なども含む。

【００４０】

複数のスレッドをグループとしてまとめたものを、スレッドグループと呼ぶ。

スレッドグループは、グループに含まれるスレッドすべてに対して同じ属性を与える、などの処理を効率よく簡単にできるメリットがある。ハードリアルタイムクラスまたはソフトリアルタイムクラスのスレッドグループは、密結合スレッドグループ（tightly coupled thread group）と疎結合スレッドグループ（loosely coupled thread group）の２種類に大別される。密結合スレッドグループ（tightly coupled thread group）と疎結合スレッドグループ（loosely coupled thread group）はスレッドグループに付加された属性情報（結合属性情報）によって識別される。アプリケーションプログラム内のコードまたは上述の構成記述によってスレッドグループの結合属性を明示的に指定することができる。

密結合スレッドグループは互いに協調して動作する複数のスレッドの集合から構成されるスレッドグループである。すなわち、密結合スレッドグループは、そのグループに属するスレッド群が、お互いに密接に連携して動作することを示す。密接な連携とは、例えば，頻繁にスレッド間で通信あるいは同期処理などの相互作用（interaction）を行ったり、あるいは、レイテンシ（latency）（遅延）の小さい相互作用を必要とする場合などである。一方、疎結合スレッドグループは、密結合スレッドグループに比べてそのグループに属するスレッド群間の密接な連携が不要であるスレッドグループであり、スレッド群はメモリ１４上のバッファを介してデータ受け渡しのための通信を行う。

【００４１】

（密結合スレッドグループ）

図２５に示すように、密結合スレッドグループに属するスレッド群にはそれぞれ別のＶＰＵが割り当てられ、各スレッドが同時に実行される。密結合スレッドグループに属するスレッドを、密結合スレッド（tightly coupled thread）と呼ぶ。この場合、密結合スレッドグループに属する密結合スレッドそれぞれの実行期間がそれら密結合スレッドの個数と同数のＶＰＵそれぞれに対して予約され、それら密結合スレッドが同時に実行される。図２５においては、ある密結合スレッドグループにスレッドＡ，Ｂの２つが密結合スレッドとして含まれており、それらスレッドＡ，ＢがそれぞれＶＰＵ０，ＶＰＵ１によって同時に実行されている様子を示している。スレッドＡ，Ｂをそれぞれ別のＶＰＵによって同時に実行

することを保証することにより、各スレッドは相手のスレッドが実行されているＶＰＵのローカルストレージや制御レジスタを通じて相手のスレッドとの通信を直接的に行うことが出来る。図２６は、スレッドＡ，Ｂがそれぞれ実行されるＶＰＵ０，ＶＰＵ１のローカルストレージを介してスレッドＡ，Ｂ間の通信が実行される様子を示している。この場合、スレッドＡが実行されるＶＰＵ０においては、そのスレッドＡのＥＡ空間の一部に、通信相手のスレッドＢが実行されるＶＰＵ１のローカルストレージ３２に対応するＲＡ空間がマッピングされる。このマッピングのためのアドレス変換は、ＶＰＵ０のメモリコントローラ３３内に設けられたアドレス変換ユニット３３１がセグメントテーブルおよびページテーブルを用いて実行する。スレッドＢが実行されるＶＰＵ１においては、そのスレッドＢのＥＡ空間の一部に、通信相手のスレッドＡが実行されるＶＰＵ０のローカルストレージ３２に対応するＲＡ空間がマッピングされる。このマッピングのためのアドレス変換は、ＶＰＵ１のメモリコントローラ３３内に設けられたアドレス変換ユニット３３１がセグメントテーブルおよびページテーブルを用いて実行する。図２７には、ＶＰＵ０上で実行されるスレッドＡが自身のＥＡ空間にスレッドＢが実行されるＶＰＵ１のローカルストレージ（ＬＳ１）３２をマッピングし、ＶＰＵ１上で実行されるスレッドＢが自身のＥＡ空間にスレッドＡが実行されるＶＰＵ０のローカルストレージ（ＬＳ０）３２をマッピングした様子が示されている。例えば、スレッドＡはスレッドＢに引き渡すべきデータがローカルストレージＬＳ０上に準備できた時点で、そのことを示すフラグをローカルストレージＬＳ０またはスレッドＢが実行されるＶＰＵ１のローカルストレージＬＳ１にセットする。スレッドＢはそのフラグのセットに応答して、ローカルストレージＬＳ０上のデータをリードする。

【００４２】

このように、結合属性情報によって密結合関係にあるスレッドを特定できるようにすると共に、結合関係にあるスレッドＡ，Ｂがそれぞれ別のＶＰＵによって同時に実行されることを保証することにより、スレッドＡ，Ｂ間の通信、同期に関するインタラクションをより軽量で且つ遅延無く行うことが可能となる。

【００４３】

（疎結合スレッドグループ）

疎結合スレッドグループに属するスレッド群それぞれの実行時間は、それらスレッド群間の入出力関係によって決定され、たとえ実行順序の制約がないスレッド同士であってもそれらが同時に実行されることは保証されない。疎結合スレッドグループ属するスレッドを、疎結合スレッド（loosely coupled thread）と呼ぶ。図２８においては、ある疎結合スレッドグループにスレッドＣ，Ｄの２つが疎結合スレッドとして含まれており、それらスレッドＣ，ＤがそれぞれＶＰＵ０，ＶＰＵ１によって実行されている様子を示している。図２８に示すように、各スレッドの実行時間はばらばらになる。スレッドＣ，Ｄ間の通信は、図２９に示すように、メインメモリ１４上に用意したバッファを介して行われる。スレッドＣはローカルストレージＬＳ０に用意したデータをＤＭＡ転送によってメインメモリ１４上に用意したバッファに書き込み、スレッドＤはその開始時にＤＭＡ転送によってメインメモリ１４上のバッファからローカルストレージＬＳ１にデータを読み込む。

【００４４】

（プロセスとスレッド）

プロセスは，図３０に示すように、一つのアドレス空間と一つ以上のスレッドから構成される。一つのプロセスに含まれるスレッドの数と種類は，どのような組み合わせでも構わない。例えば，ＶＰＵスレッドのみから構成されるプロセスも構築可能であるし，ＶＰＵスレッドとＭＰＵスレッドが混在するプロセスも構築可能である。スレッドがスレッド固有の情報としてスレッドコンテクストを保持しているのと同様に，プロセスもプロセス固有の情報としてプロセスコンテクストを保持する。このプロセスコンテクストには，プロセスに固有であるアドレス空間と，プロセスが含んでいる全スレッドのスレッドコンテクストが含まれる。プロセスのアドレス空間は，プロセスに属するすべてのスレッド間で共有することができる。一つのプロセスは，複数のスレッドグループを含むことができる。しかし，一つのスレッドグループが複数のプロセスに属することはできない。このため，あるプロセスに属するスレッドグループは，そのプロセスに固有であるということになる。本実施形態のリアルタイム処理システムにおいて、スレッ

ドを新しく生成する方式には、Thread first modelとAddress space first modelの２種類がある。Address space first modelは既存のＯＳで採用されているのと同様の方式で、ＭＰＵスレッドにもＶＰＵスレッドにも適用できる。一方、Thread first modelはＶＰＵスレッドにしか適用できない方式で、本発明のリアルタイム処理システムに特有の方式である。Thread first modelでは，既存のスレッド（新しくスレッドを作りたいと思っている側のスレッド。新しく作るスレッドの親になるスレッドのこと。）は，まず新規スレッドが実行するプログラムを指定して，新規スレッドにプログラムの実行を開始させる。この時、プログラムはＶＰＵ１２のローカルストレージに格納され、所定の実行開始番地から処理が開始される。この時点では、この新規スレッドにはアドレス空間が関連付けられていないので、自身のローカルストレージはアクセスできるが、メモリ１４はアクセスできない。その後，新規スレッドは，必要に応じて自身でＶＰＵ実行環境のサービスを呼び出してアドレス空間を生成して関連付けたり、ＭＰＵ１１側の処理によってアドレス空間を関連付けられたりして、メモリ１４にアクセスできるようになる。Address space first modelでは，既存のスレッドは，新しくアドレス空間を生成するか、あるいは既存のアドレス空間を指定して、そのアドレス空間に新規スレッドが実行するプログラムを配置する。そして新規スレッドにそのプログラムの実行を開始させる。Thread first modelのメリットは、ローカルストレージだけで動作するので、スレッドの生成やディスパッチや終了処理などのオーバーヘッドを小さくできることである。

【００４５】

（スレッド群のスケジューリング）

次に、図３１のフローチャートを参照して、ＶＰＵ実行環境４０１によって実行されるスケジューリング処理について説明する。ＶＰＵ実行環境４０１内のスケジューラは、スケジュール対象のスレッド群にスレッドグループ単位で付加されている結合属性情報に基づいて、スレッド間の結合属性をチェックし（ステップＳ１２１）、各スレッドグループ毎にそのスレッドグループが密結合スレッドグループおよび疎結合スレッドグループのいずれであるかを判別する（ステップＳ１２２）。結合属性のチェックは、プログラムコード中のスレッドに関する記

述あるいは上述の構成記述１１７中のスレッドパラメータを参照することによって行われる。このようにして、密結合スレッドグループおよび疎結合スレッドグループをそれぞれ特定することにより、スケジュール対象のスレッド群は密結合スレッドグループと疎結合スレッドグループとに分離される。

【００４６】

密結合スレッドグループに属するスレッド群に対するスケジューリングは次のように行われる。すなわち、ＶＰＵ実行環境４０１内のスケジューラは、スケジュール対象のスレッド群から選択された密結合スレッドグループに属するスレッド群がそれぞれ別のＶＰＵによって同時に実行されるように、その密結合スレッドグループに属するスレッド群と同数のＶＰＵそれぞれの実行期間を予約し、スレッド群をそれら予約したＶＰＵそれぞれに同時にディスパッチする（ステップＳ１２３）。そして、スケジューラは、各スレッドが実行されるＶＰＵ内のアドレス変換ユニット３３１を用いて、各スレッドのＥＡ空間の一部に、協調して相互作用を行う相手となる他のスレッドが実行されるＶＰＵのローカルストレージに対応するＲＡ空間をマッピングする（ステップＳ１２４）。一方、スケジュール対象のスレッド群から選択された疎結合スレッドグループに属する疎結合スレッド群については、スケジューラは、それらスレッド群間の入出力関係に基づいてそれらスレッド群を１以上のＶＰＵに順次ディスパッチする（ステップＳ１２５）。

【００４７】

（ローカルストレージのマッピング）

本実施形態のリアルタイム処理システムにおいて、ＭＰＵスレッドとＶＰＵスレッドの間、あるいはＶＰＵスレッドと他のＶＰＵスレッドの間で、何らかの通信や同期を行いながら協調して動作を行う場合には、協調相手のＶＰＵスレッドのローカルストレージにアクセスする必要がある。たとえば、より軽量で高速な同期機構は、ローカルストレージ上に同期変数を割り付けて実装する。そのため、あるＶＰＵ１２のローカルストレージを、他のＶＰＵ１２あるいはＭＰＵ１１のスレッドが直接アクセスする必要がある。図４に示す例のように、各ＶＰＵ１２のローカルストレージが実アドレス空間に割り付けられている場合、セグメン

トテーブルやページテーブルを適切に設定すれば、相手のＶＰＵ１２のローカルストレージを直接アクセスすることができる。しかしこの場合に、大きく２つの問題が生じる。

【００４８】

第１の問題は、ＶＰＵスレッドのディスパッチ先ＶＰＵ１２の変更に関する問題である。図３２のように，ＶＰＵスレッドＡとＢが存在し，それぞれＶＰＵ０とＶＰＵ１で動いているとする。そして，このスレッドＡとＢはお互いのスレッドと協調したいので，お互いのスレッドのＬＳ（ローカルストレージ）を，自分のＥＡ空間にマッピングしているとする。また，ＶＰＵ０，１，２のＬＳ０，１，２はそれぞれ図３２のようにＲＡ空間に存在するとする。この時，ＶＰＵスレッドＡが，自分のＥＡ空間にマッピングしているのは，ＶＰＵスレッドＢが動いているＶＰＵのＬＳ，つまり，ＶＰＵ１のＬＳであるＬＳ１である。逆に，ＶＰＵスレッドＢが，自分のＥＡ空間にマッピングしているのは，ＶＰＵスレッドＡが動いているＶＰＵのＬＳ，つまり，ＶＰＵ０のＬＳであるＬＳ０である。その後，ＶＰＵ実行環境の中のスケジューラによって、ＶＰＵスレッドＡを実行するＶＰＵがディスパッチされて，ＶＰＵスレッドＡは，ＶＰＵ２で動くことになったとする。この時、もはやＶＰＵスレッドＡはＶＰＵ０では動いていないので，ＶＰＵスレッドＢが，自分のＥＡ空間にマッピングしているＶＰＵ０のＬＳは，意味がなくなる。この場合，スレッドＢが，スレッドＡのディスパッチ先ＶＰＵが変更になったことを知らなくてもいいように，システムは何らかの方法でＬＳ０にマッピングされていたＥＡ空間のアドレスをＬＳ２にマッピングして、スレッドＢから、スレッドＡのローカルストレージとしてＶＰＵ２のＬＳであるＬＳ２が見えるようにする必要がある。

【００４９】

第２の問題は、物理ＶＰＵと論理ＶＰＵの対応関係の問題である。ＶＰＵをＶＰＵスレッドに割り当てるまでには，実際には，２つのレベルがある。一つは論理ＶＰＵのＶＰＵスレッドへの割り当てであり，もう一つが物理ＶＰＵの論理ＶＰＵへの割り当てである。物理ＶＰＵとは，仮想計算機ＯＳ３０１が管理している物理的なＶＰＵ１２である。そして，論理ＶＰＵとは，仮想計算機ＯＳ３０１

がゲストＯＳ割り当てた、論理的なＶＰＵのことである。この関係は図１４にも示している。たとえば、ＶＰＵ実行環境４０１が論理的なＶＰＵを管理する場合、図３２の例で、ＶＰＵスレッドの割り当て対象となるＶＰＵは論理ＶＰＵである。

【００５０】

図３３は，この２つのレベルの割り当ての概念を示している。直前に説明した第１の問題は，図３３の上の段に位置する，ＶＰＵスレッドの論理ＶＰＵへの割り当て問題に相当する。第２の問題である物理ＶＰＵの論理ＶＰＵへの割り当て問題は，下の段に位置する割り当てに相当する。図３３では，４つの物理ＶＰＵから，３つのＶＰＵを選択し，３つの論理ＶＰＵに割り当てていることを示している。もし，この物理ＶＰＵと論理ＶＰＵの対応関係が変わった場合，ＶＰＵスレッドの論理ＶＰＵへの割り当てが変更になっていなくても，適切な設定の変更が必要となる。例えば，変更後の論理ＶＰＵのＬＳに対するアクセスが，正しい物理ＶＰＵのＬＳを指すように，ＬＳに対応するページテーブルエントリを入れ換える，などである。

【００５１】

ある時刻に，図３４のように，物理ＶＰＵ１，２，３が論理ＶＰＵ０，１，２にそれぞれ割り当てられているとする。そして，論理ＶＰＵ１はＶＰＵスレッドＡに，そして論理ＶＰＵ２はＶＰＵスレッドＢに割り当てられていたとする。そして，ＶＰＵスレッドＡとＢは，それぞれ，お互いに，相手の動作している物理ＶＰＵのＬＳを自分のＥＡ空間にマッピングしているとする。ＶＰＵスレッドＡのＥＡ空間にはＶＰＵスレッドＢが実行されている物理ＶＰＵ３のＬＳ３が，そしてＶＰＵスレッドＢのＥＡ空間にはＶＰＵスレッドＡが実行されている物理ＶＰＵ２のＬＳ２がマッピングされている。その後，ある時刻に，仮想計算機ＯＳ３０１によって、論理ＶＰＵ０，１が物理ＶＰＵ０，１に，再割り当てされたとする。すると，今までＶＰＵスレッドＡが動作していた論理ＶＰＵ１は，物理ＶＰＵ２から物理ＶＰＵ１へと変化する。論理ＶＰＵのＶＰＵスレッドへの割り当ては変化していないが，物理ＶＰＵと論理ＶＰＵの対応関係が変化したことになる。このため，ＶＰＵスレッドＢがＥＡ空間にマッピングしている，ＶＰＵスレ

ッドＡの動作しているＶＰＵのＬＳを，ＬＳ２からＬＳ１に変更し，正しくアクセスできるようにする必要がある。

【００５２】

これらの２つの問題を解決するために、本実施形態のリアルタイム処理システムでは、スレッドから見たＥＡ空間の固定アドレスに、必ず相手のスレッドを実行しているＶＰＵのローカルストレージがマップされて見えるように仮想記憶機構を制御する。すなわち、ＶＰＵスケジューラによる論理ＶＰＵのディスパッチ時、および仮想計算機ＯＳ等による物理ＶＰＵと論理ＶＰＵの対応関係の切り替え時に、適宜ページテーブルやセグメントテーブルを書き換えることで、ＶＰＵ上で動作しているスレッドからは、いつも同じ番地に相手のスレッドを実行しているＶＰＵのローカルストレージが見えるようにする。

【００５３】

まず、２つのスレッドのＥＡ空間の関係について説明する、２つのスレッドのＥＡ空間は、次の３つのいずれかのパターンで共有あるいは非共有になっている。

１．共有ＥＡ型：　２つのスレッド１，２がセグメントテーブルもページテーブルも共有している（図３５）

２．共有ＶＡ型：　２つのスレッド１，２は、ページテーブルは共有するが、セグメントテーブルは共有せず、それぞれが持っている（図３６）

３．非共有型：　２つのスレッド１，２はページテーブルもセグメントテーブルも共有せず、それぞれが持っている（図３７）

以下、１の共有ＥＡ型を例に、ＶＰＵのローカルストレージをどのようにマップするように制御するかについて説明する。

まず、図３８に示すように、ＶＡ空間上に各論理ＶＰＵに対応した領域を設け、そこに、その論理ＶＰＵが対応付けられている物理ＶＰＵのローカルストレージがマップされるように、ページテーブルを設定する。この例の場合、物理ＶＰＵ０，１，２がそれぞれ論理ＶＰＵ０，１，２に対応付けられている状態を示している。次に、スレッドＡからはスレッドＢを実行しているＶＰＵのローカルストレージが、固定アドレスであるセグメントａの領域に見えるように、セグメン

トテーブルを設定する。また、スレッドＢからはスレッドＡを実行している論理ＶＰＵのローカルストレージが、固定アドレスであるセグメントｂに見えるように、セグメントテーブルを設定する。この例では、スレッドＡは論理ＶＰＵ２で、スレッドＢは論理ＶＰＵ１で実行している状況を示している。ここで、ＶＰＵ実行環境４０１のスケジューラが、スレッドＢを論理ＶＰＵ０にディスパッチしたとする。この時、ＶＰＵ実行環境４０１は、図３９に示すように、スレッドＡからは固定アドレスであるセグメントａを通して、スレッドＢを現在実行している論理ＶＰＵ０のローカルストレージを見えるように、ＶＰＵ実行環境４０１はセグメントテーブルを自動的に書き換える。

さらにここで、たとえば仮想計算機ＯＳ３０１がゲストＯＳのディスパッチをしたため、物理ＶＰＵと論理ＶＰＵの対応が変化したとする。このとき、たとえば図４０に示すように、ＶＰＵ実行環境４０１は、ページテーブルを書き換えて、ＶＡ空間上に固定されている論理ＶＰＵのローカルストレージの領域が、正しい物理ＶＰＵのローカルストレージの領域を指すようにする。図４０の例では、物理ＶＰＵ１，２，３が論理ＶＰＵ０，１，２に対応するように変更されたため、ページテーブルを書き換えて、現在の正しいマッピングになるようにしている。

【００５４】

このように、ＶＰＵ実行環境４０１のスケジューラのディスパッチによって、スレッドを実行する論理ＶＰＵが変更になった場合には、ＥＡ空間からＶＡ空間へのマッピングを行っているセグメントテーブルを書き換えて、第１の問題を解決している。また、仮想計算機ＯＳ３０１などによって、物理ＶＰＵと論理ＶＰＵの対応が変更になった場合は、ＶＡ空間からＲＡ空間へのマッピングを行っているページテーブルを書き換えて、第２の問題を解決している。

このようして、相互作用を行う相手のスレッドが実行されるプロセッサに応じて、実効アドレス空間にマッピングされる、相手のスレッドに対応するプロセッサのローカルメモリが自動的に変更することにより、各スレッドは相手のスレッドがディスパッチされるプロセッサを意識することなく、相手のスレッドとの相互作用を効率よく行うことが出来る。よって、複数のスレッドを効率よく並列に

実行することが可能となる。

【００５５】

以上、共有ＥＡ型の場合の例を説明したが、２の共有ＶＡ型、３の非共有型についても、セグメントテーブルまたはページテーブルを書き換えることにより、同様にして第１の問題および第２の問題を解決することができる。

【００５６】

上記の第１および第２の問題を解決する別の方法について述べる。ここでも、共有ＥＡ型の場合を例に説明する。図４１に示すように、協調して動作する複数のＶＰＵスレッドがある場合、それらのスレッドを実行するＶＰＵのローカルストレージを、セグメント上に連続してマップするように、ページテーブルとセグメントテーブルを設定する。図４１の例の場合、スレッドＡは物理ＶＰＵ２で、スレッドＢは物理ＶＰＵ０で実行されており、それぞれのＶＰＵのローカルストレージが同一のセグメントに連続して配置されるように、ページテーブルとセグメントテーブルを設定している。ここで、ＶＰＵ実行環境４０１のスケジューラによってスレッドを実行する論理ＶＰＵがディスパッチされたり、仮想計算機ＯＳ３０１等によって物理ＶＰＵと論理ＶＰＵの対応が変更になった場合には、それぞれの変更がスレッドＡおよびスレッドＢに対して隠蔽されるように、ページテーブルを書き換えて、ＶＡ空間とＲＡ空間のマップを変更する。たとえば図４２は、スレッドＡを実行しているＶＰＵが物理ＶＰＵ１に、スレッドＢを実行しているＶＰＵが物理ＶＰＵ３に変更になった場合のマッピングを示している。この変更が行われても、スレッドＡおよびスレッドＢからは、固定したアドレスを持つセグメント内の、所定の領域をアクセスすることで、常に相手のスレッドを実行しているＶＰＵのローカルストレージをアクセスすることができる。

【００５７】

次に、図４３のフローチャートを参照して、ＶＰＵ実行環境４０１によって実行されるアドレス管理処理の手順について説明する。ＶＰＵ実行環境４０１は、各スレッドのＥＡ空間上の固定アドレスに、相手スレッドを実行しているＶＰＵのローカルストレージに対応するＲＡ空間をマッピングする（ステップＳ２０１）。この後、ＶＰＵ実行環境４０１は、相手スレッドのディスパッチ先ＶＰＵの

変更あるいは論理ＶＰＵと物理ＶＰＵの対応関係の変更に起因して、相手スレッドが実行されるＶＰＵが変更されたかどうかを判別する（ステップＳ２０２）。相手スレッドが実行されるＶＰＵが変更されたならば、ＶＰＵ実行環境４０１は、セグメントテーブルまたはページテーブルの内容を書き換えて、各スレッドのＥＡ空間上の固定アドレスにマッピングされているローカルストレージを、相手スレッドが実行されるＶＰＵに合わせて変更する（ステップＳ２０３）。

【００５８】

これまでの例では、蜜結合スレッドグループのように、互いにＶＰＵによって実行中のスレッド間で、相手のスレッドを実行しているＶＰＵのローカルストレージをアクセスする方式を説明した。しかし、疎結合スレッドグループなど、協調して動作するスレッドが必ずしも同時にＶＰＵに割り当てられて実行していない場合も存在する。そのような場合でも、ＥＡ空間上には相手のスレッドを実行しているＶＰＵ１２のローカルストレージをマップする領域は存在するので、その領域を以下のように用いて対処する。

【００５９】

第１の方法：　相手のスレッドが実行中で無い場合には、そのスレッドに対応するＶＰＵのローカルストレージをマップする領域にアクセスすると、スレッドは相手のスレッドが実行開始するまで待たされるようにする。

第２の方法：　相手のスレッドが実行中で無い場合には、そのスレッドに対応するＶＰＵのローカルストレージをマップする領域にアクセスすると、スレッドは例外発生やエラーコードによって、その旨を知る。

【００６０】

第３の方法：　スレッドの終了時に、そのスレッドを最後に実行していたときのローカルストレージの内容をメモリに保存しておき、そのスレッドに対応付けられたローカルストレージを指すページテーブルあるいはセグメントテーブルのエントリからは、そのメモリ領域を指すようにマッピングを制御する。この方式により、相手のスレッドが実行中でなくても、相手のスレッドに対応付けられたローカルストレージがあたかもあるように、スレッドの実行を続けることができる。図４４および図４５に具体例を示す。

①：　いま、スレッドＡ，ＢがそれぞれＶＰＵ０，１で実行されており、スレッドＢのＥＡ空間には相手のスレッドＡが実行されているＶＰＵ０のローカルストレージＬＳ０がマッピングされているとする。

②：　スレッドＡの終了時には、スレッドＡまたはＶＰＵ実行環境４０１は、スレッドＡが実行されているＶＰＵ０のローカルストレージＬＳ０の内容をメモリ１４に保存する（ステップＳ２１１）。

③：　ＶＰＵ実行環境４０１は、スレッドＢのＥＡ空間にマッピングされている相手先スレッドＡのローカルストレージのアドレス空間を、ＶＰＵ０のＬＳ０から、ＬＳ０の内容が保存されたメモリ１４上のメモリ領域に変更する（ステップＳ２１２）。これにより、スレッドＢは、相手のスレッドＡが実行中でなくなった後も、その動作を継続することができる。

④：　スレッドＡに再びＶＰＵが割り当てられたとき、ＶＰＵ実行環境４０１は、メモリ１４上のメモリ領域をスレッドＡが実行されるＶＰＵのローカルストレージに戻す（ステップＳ２１３）。たとえばスレッドＡに再びＶＰＵ０が割り当てられたときは、メモリ１４上のメモリ領域の内容は、ＶＰＵ０のローカルストレージＬＳ０に戻される。

⑤：　ＶＰＵ実行環境４０１は、スレッドＢのＥＡ空間にマッピングされている相手先スレッドＡのローカルストレージのアドレス空間を、スレッドＡが実行されるＶＰＵのローカルストレージに変更する（ステップＳ２１４）。たとえばスレッドＡに再びＶＰＵ０が割り当てられたときは、スレッドＢのＥＡ空間にマッピングされている相手先スレッドＡのローカルストレージのアドレス空間は、ＶＰＵ０のローカルストレージＬＳ０に戻される。

【００６１】

なお、スレッドＡにＶＰＵ２が割り当てられたときは、メモリ１４上のメモリ領域の内容は、ＶＰＵ２のローカルストレージＬＳ２に復元される。そして、スレッドＢのＥＡ空間にマッピングされている相手先スレッドＡのローカルストレージのアドレス空間は、ＶＰＵ２のローカルストレージＬＳ２に変更される。

【００６２】

なお、図１の計算機システムに設けられたＭＰＵ１１と複数のＶＰＵ１２は。

それらを１チップ上に混載した並列プロセッサとして実現することもできる。この場合も、ＭＰＵ１１によって実行されるＶＰＵ実行環境、あるいは特定の一つのＶＰＵなどによって実行されるＶＰＵ実行環境が、複数のＶＰＵ１２に対するスケジューリングおよびアドレス管理を行うことが出来る。

【００６３】

またＶＰＵ実行環境として動作するプログラムまたはそのＶＰＵ実行環境を含むオペレーティングシステムなどのプログラムをコンピュータ読み取り可能な記憶媒体に記憶することにより、その記憶媒体を通じて当該プログラムを、ローカルプロセッサをそれぞれ有する複数のプロセッサを含むコンピュータに導入して実行するだけで、本実施形態と同様の効果を得ることが出来る。

【００６４】

また、本発明は上記実施形態そのままに限定されるものではなく、実施段階ではその要旨を逸脱しない範囲で構成要素を変形して具体化できる。また、上記実施形態に開示されている複数の構成要素の適宜な組み合わせにより、種々の発明を形成できる。例えば、実施形態に示される全構成要素から幾つかの構成要素を削除してもよい。さらに、異なる実施形態にわたる構成要素を適宜組み合わせてもよい。

【００６５】

【発明の効果】

以上説明したように、本発明によれば、複数のプロセッサを用いて複数のスレッドを効率よく並列に実行することが可能となる。

【図面の簡単な説明】

【図１】　本発明の一実施形態に係るリアルタイム処理システムを構成する計算機システムの例を示すブロック図。

【図２】　同実施形態のリアルタイム処理システムに設けられたＭＰＵおよびＶＰＵそれぞれの構成を示すブロック図。

【図３】　同実施形態のリアルタイム処理システムで用いられる仮想アドレス変換機構の例を示す図。

【図４】　同実施形態のリアルタイム処理システムにおける実アドレス空間

にマッピングされるデータの例を示す図。

【図５】　同実施形態のリアルタイム処理システムにおける実効アドレス空間、仮想アドレス空間、実アドレス空間を説明するための図。

【図６】　デジタルテレビ放送の受信機の構成を示すブロック図。

【図７】　同実施形態のリアルタイム処理システムによって実行されるプログラムモジュールの構成の例を示す図。

【図８】　図７のプログラムモジュール内に含まれる構成記述の例を示す図。

【図９】　図７のプログラムモジュールに対応するプログラム間のデータの流れを示す図。

【図１０】　図７のプログラムモジュールが２つのＶＰＵによって並列に実行される様子を示す図。

【図１１】　図７のプログラムモジュールが２つのＶＰＵによってパイプライン形式で実行される様子を示す図。

【図１２】　同実施形態のリアルタイム処理システムにおけるオペレーティングシステムの実装形態の例を示す図。

【図１３】　同実施形態のリアルタイム処理システムにおけるオペレーティングシステムの実装形態の他の例を示す図。

【図１４】　同実施形態のリアルタイム処理システムにおける仮想計算機ＯＳとゲストＯＳとの関係を示す図。

【図１５】　同実施形態のリアルタイム処理システムにおいて複数のゲストＯＳに時分割で資源が割り当てられる様子を示す図。

【図１６】　同実施形態のリアルタイム処理システムにおいてある特定のゲストＯＳによって特定の資源が専有される様子を示す図。

【図１７】　同実施形態のリアルタイム処理システムにおいてスケジューラとして用いられるＶＰＵ実行環境を示す図。

【図１８】　同実施形態のリアルタイム処理システムで用いられる仮想計算機ＯＳにＶＰＵ実行環境を実装した例を示す図。

【図１９】　同実施形態のリアルタイム処理システムで用いられる一つのゲ

ストＯＳとしてＶＰＵ実行環境を実装する例を示す図。

【図２０】　同実施形態のリアルタイム処理システムで用いられる複数のゲストＯＳそれぞれにＶＰＵ実行環境を実装する例を示す図。

【図２１】　同実施形態のリアルタイム処理システムで用いられる一つのゲストＯＳにＶＰＵ実行環境を実装する例を示す図。

【図２２】　同実施形態のリアルタイム処理システムで用いられるＭＰＵ側ＶＰＵ実行環境とＶＰＵ側ＶＰＵ実行環境を説明するための図。

【図２３】　同実施形態のリアルタイム処理システムで用いられるＶＰＵ側ＶＰＵ実行環境によって実行される処理手順を示すフローチャート。

【図２４】　同実施形態のリアルタイム処理システムで用いられるＭＰＵ側ＶＰＵ実行環境によって実行される処理手順を示すフローチャート。

【図２５】　同実施形態のリアルタイム処理システムにおいて密結合スレッドグループに属するスレッド群がそれぞれ別のプロセッサによって同時に実行される様子を示す図。

【図２６】　同実施形態のリアルタイム処理システムにおける密結合スレッド間の相互作用を説明するための図。

【図２７】　同実施形態のリアルタイム処理システムにおいて各密結合スレッドの実効アドレス空間に相手のスレッドが実行されるＶＰＵのローカルストレージがマッピングされる様子を示す図。

【図２８】　同実施形態のリアルタイム処理システムにおける疎結合スレッドグループに属するスレッド群に対するプロセッサの割り当てを説明するための図。

【図２９】　同実施形態のリアルタイム処理システムにおける疎結合スレッド間の相互作用を説明するための図。

【図３０】　同実施形態のリアルタイム処理システムにおけるプロセスとスレッドとの関係を説明するための図。

【図３１】　同実施形態のリアルタイム処理システムにおけるスケジューリング処理の手順を示すフローチャート。

【図３２】　同実施形態のリアルタイム処理システムにおけるローカルスト

レージのマッピングに関する第１の問題を説明するための図。

【図３３】　同実施形態のリアルタイム処理システムにおける物理ＶＰＵと論理ＶＰＵとの関係を示す図。

【図３４】　同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピングに関する第２の問題を説明するための図。

【図３５】　同実施形態のリアルタイム処理システムにおける実効アドレス空間共有モデルを示す図。

【図３６】　同実施形態のリアルタイム処理システムにおける仮想アドレス空間共有モデルを示す図。

【図３７】　同実施形態のリアルタイム処理システムにおける非共有モデルを示す図。

【図３８】　同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピング変更を説明するための第１の図。

【図３９】　同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピング変更を説明するための第２の図。

【図４０】　同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピング変更を説明するための第３の図。

【図４１】　同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピング変更を説明するための第４の図。

【図４２】　同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピング変更を説明するための第５の図。

【図４３】　同実施形態のリアルタイム処理システムにおいてローカルストレージのマッピング変更を行うために実行されるアドレス管理処理の手順を示すフローチャート。

【図４４】　同実施形態のリアルタイム処理システムにおいて実行されるローカルストレージとメモリとの間のマッピング変更を説明するための図。

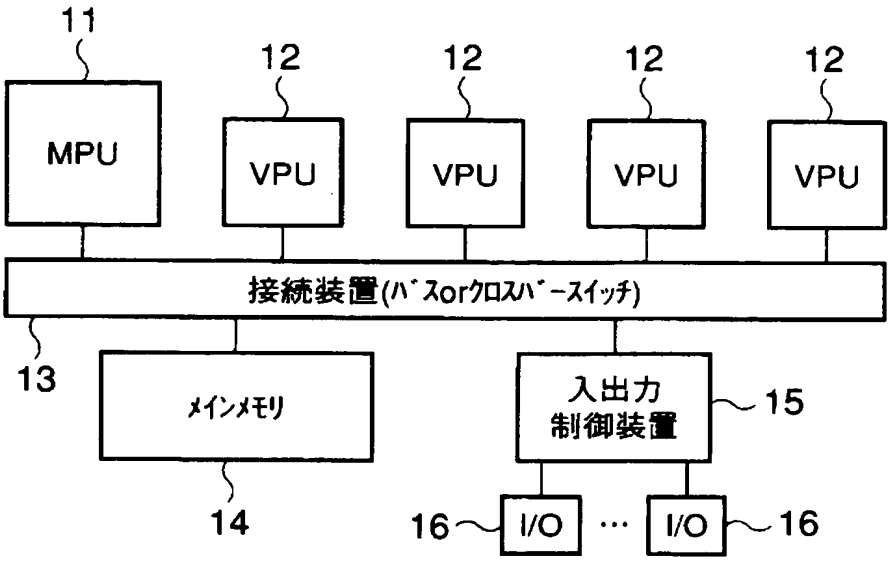
【図４５】　同実施形態のリアルタイム処理システムにおいて実行されるローカルストレージとメモリとの間のマッピング変更処理の手順を示すフローチャート。

【符号の説明】

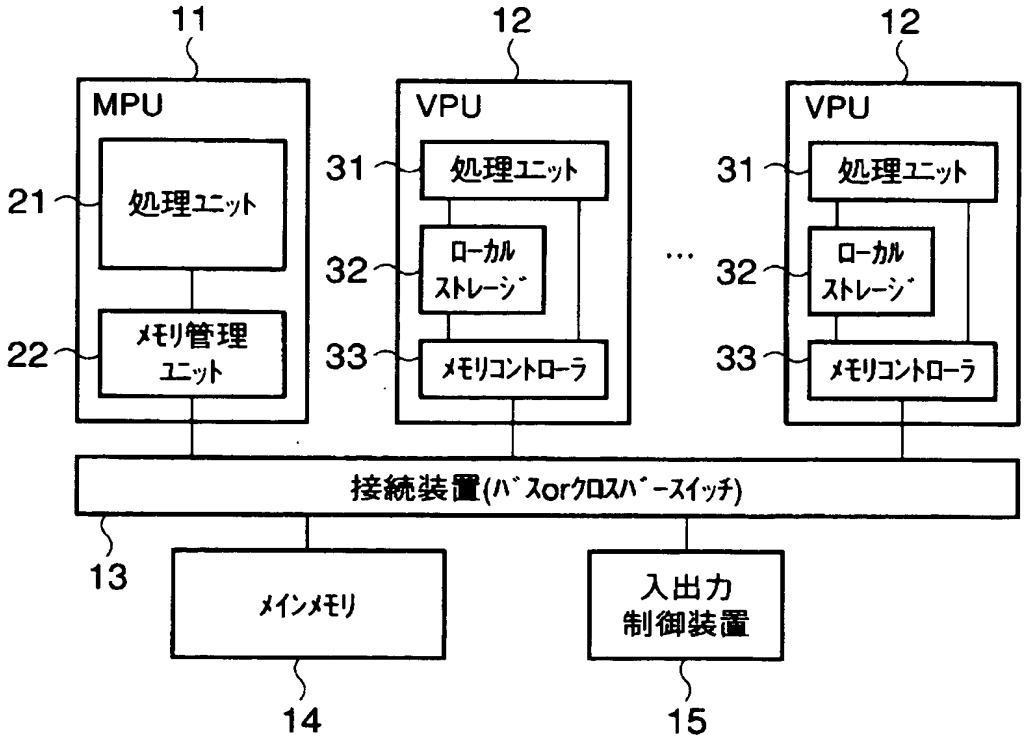
１１…ＭＰＵ（Master Processing Unit）、１２…ＶＰＵ（Slave Processing Unit）、１４…メインメモリ、２１…処理ユニット、２２…メモリ管理ユニット、３１…処理ユニット、３２…ローカルストレージ、３３…メモリコントローラ、５０…セグメントテーブル、６０…ページテーブル、１００…プログラムモジュール、１１７…構成記述、３３１…アドレス変換ユニット、４０１…ＶＰＵ実行環境。

【書類名】　図面

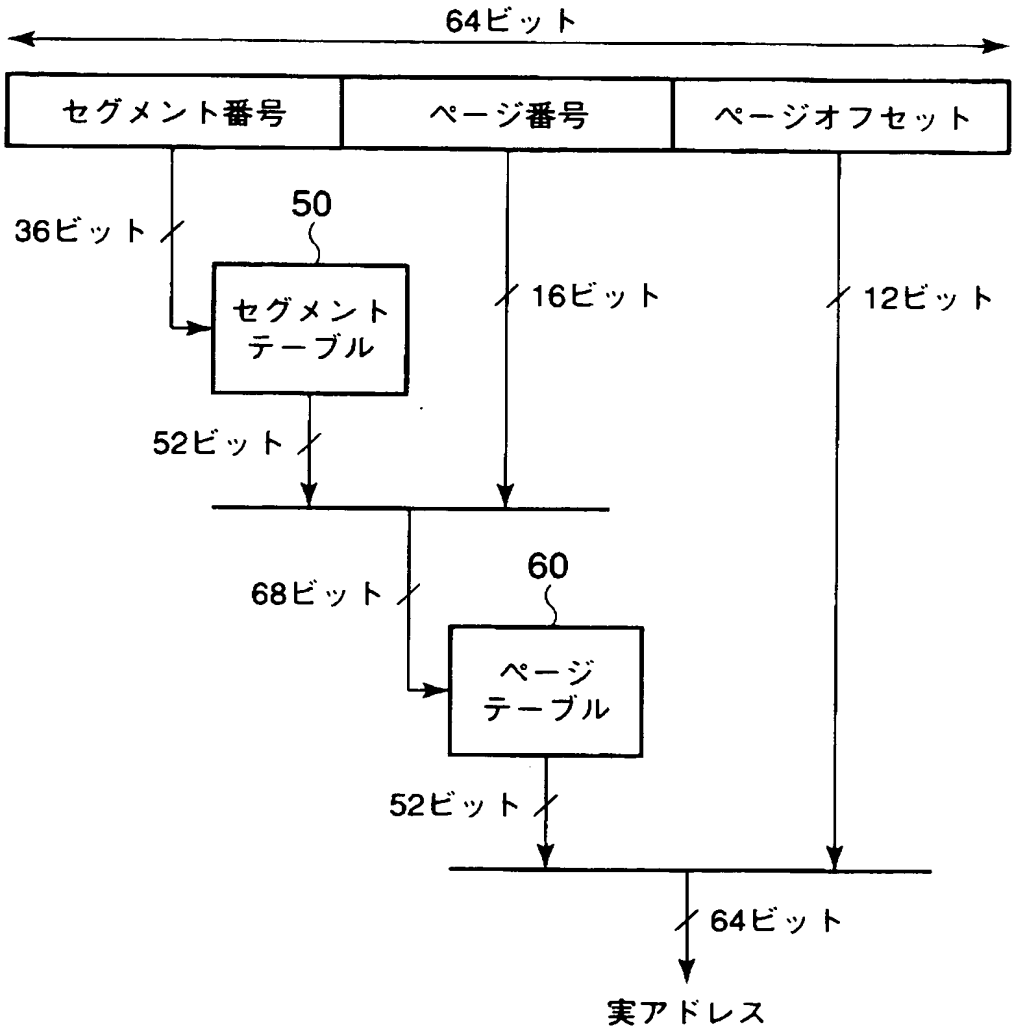
【図１】


11
12
12
12
12
MPU
VPU
VPU
VPU
VPU
接続装置(ﾊﾞｽorｸﾛｽﾊﾞｰｽｲｯﾁ)
13
ﾒｲﾝﾒﾓﾘ
入出力
制御装置
15
14
16
I/O
…
I/O
16

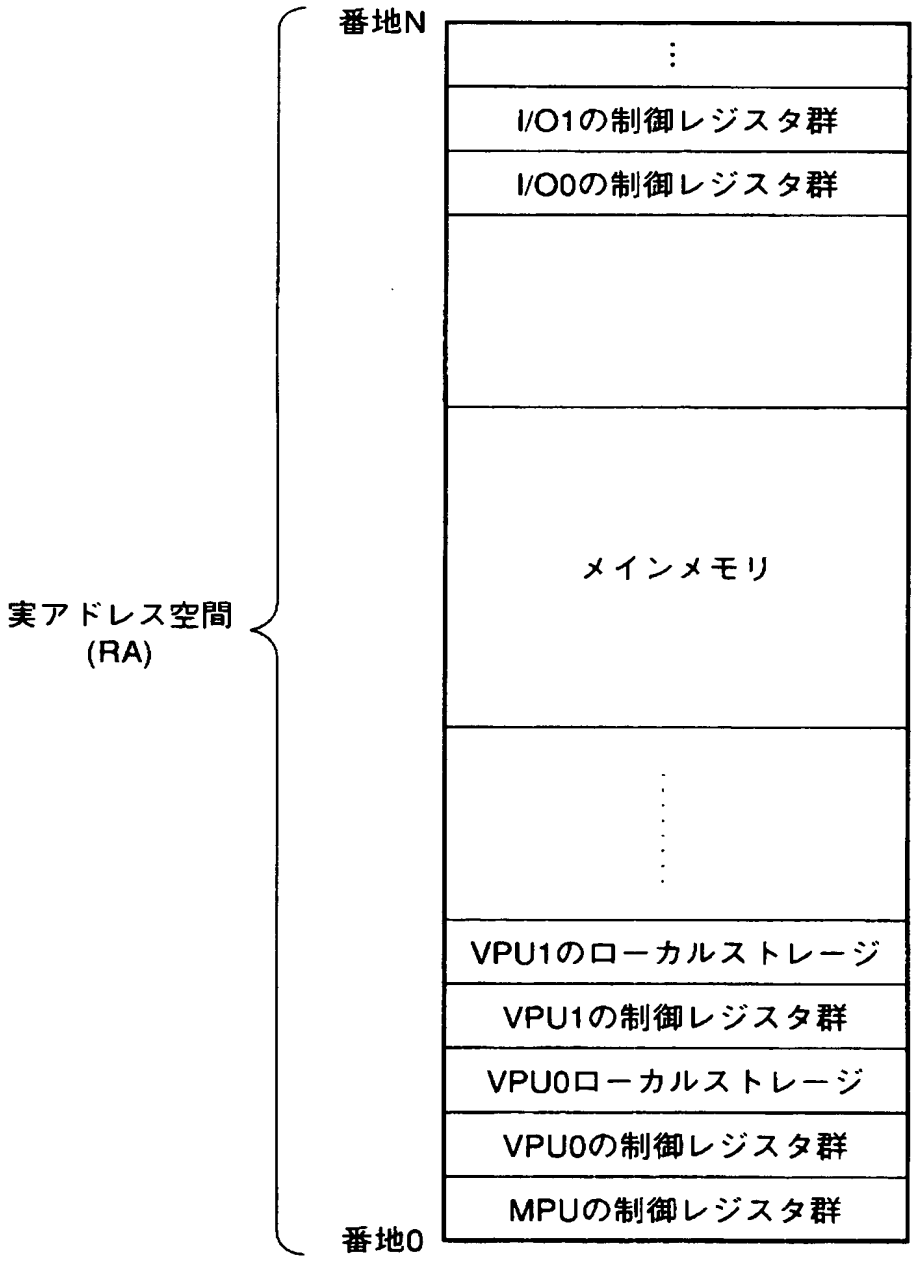
【図２】


11
12
12
MPU
VPU
VPU
処理ユニット
21
31
処理ユニット
31
処理ユニット
ローカル
ストレージ
32
ローカル
ストレージ
…
32
ローカル
ストレージ
メモリ管理
ユニット
22
33
メモリコントローラ
33
メモリコントローラ
接続装置(ﾊﾞｽorｸﾛｽﾊﾞｰｽｲｯﾁ)
13
ﾒｲﾝﾒﾓﾘ
入出力
制御装置
14
15

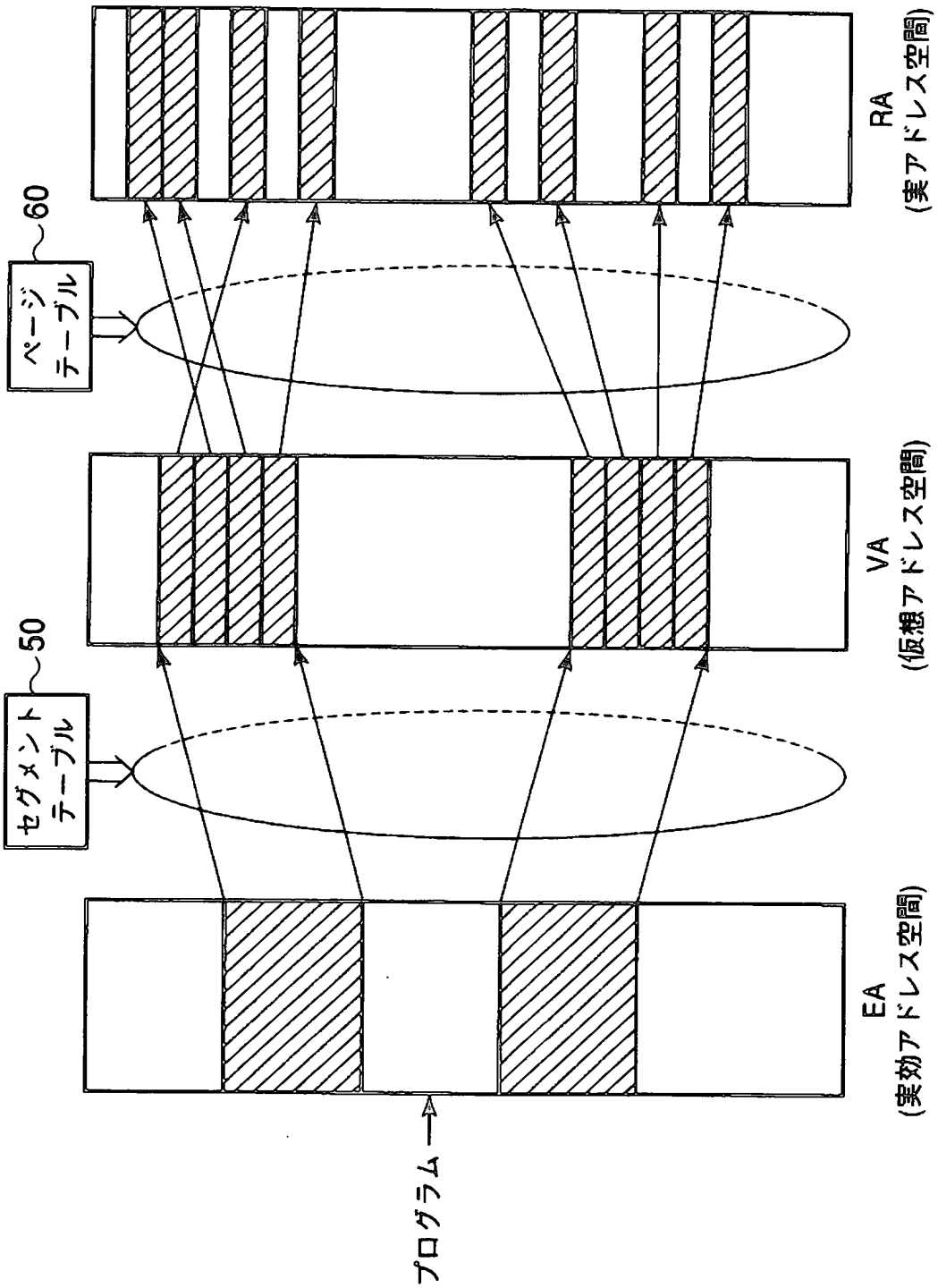
【図３】


64ビット
セグメント番号
ページ番号
ページオフセット
36ビット
50
セグメント
テーブル
16ビット
12ビット
52ビット
68ビット
60
ページ
テーブル
52ビット
64ビット
実アドレス

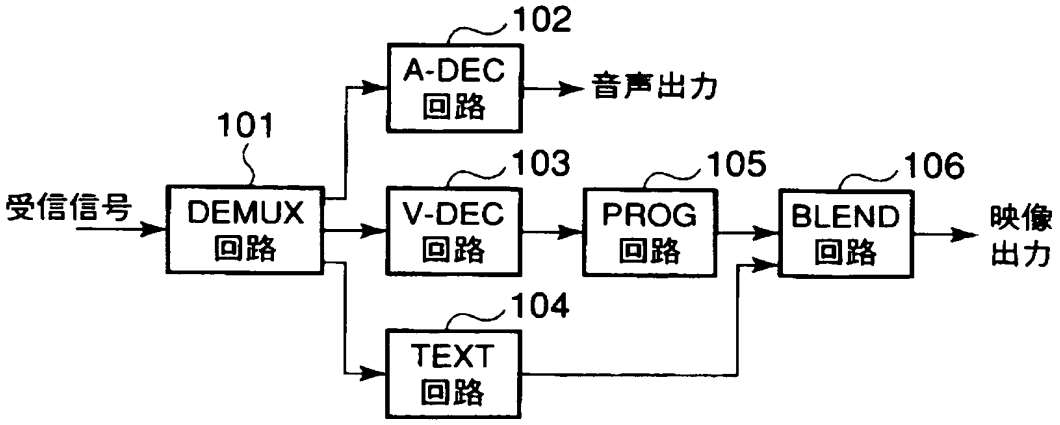
【図４】


番地N
⋮
I/O1の制御レジスタ群
I/O0の制御レジスタ群
メインメモリ
VPU1のローカルストレージ
VPU1の制御レジスタ群
VPU0ローカルストレージ
VPU0の制御レジスタ群
MPUの制御レジスタ群
番地0
実アドレス空間
(RA)

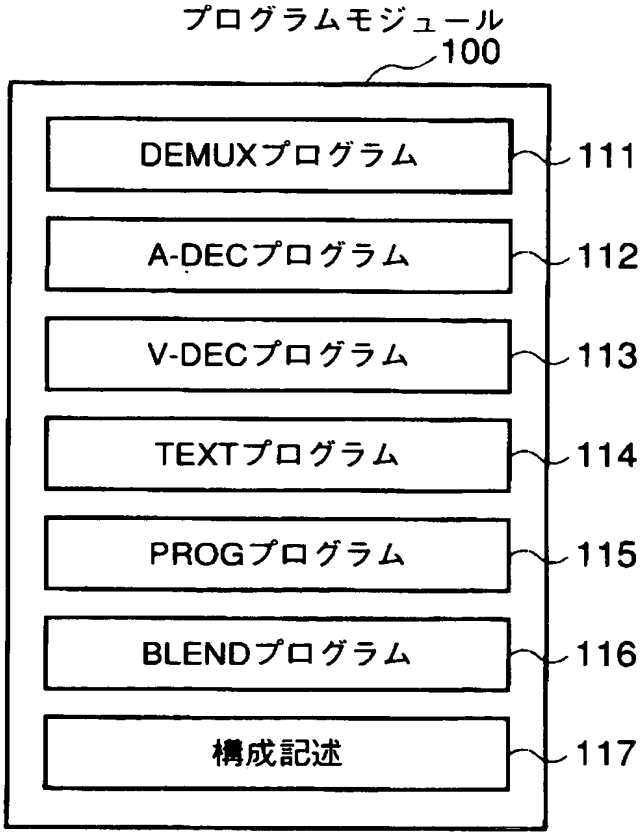
【図５】


RA
(実アドレス空間)
60
ページテーブル
VA
(仮想アドレス空間)
50
セグメントテーブル
EA
(実効アドレス空間)
プログラム

【図６】


102
A-DEC
回路
音声出力
101
受信信号
DEMUX
回路
103
V-DEC
回路
105
PROG
回路
106
BLEND
回路
映像
出力
104
TEXT
回路

【図７】

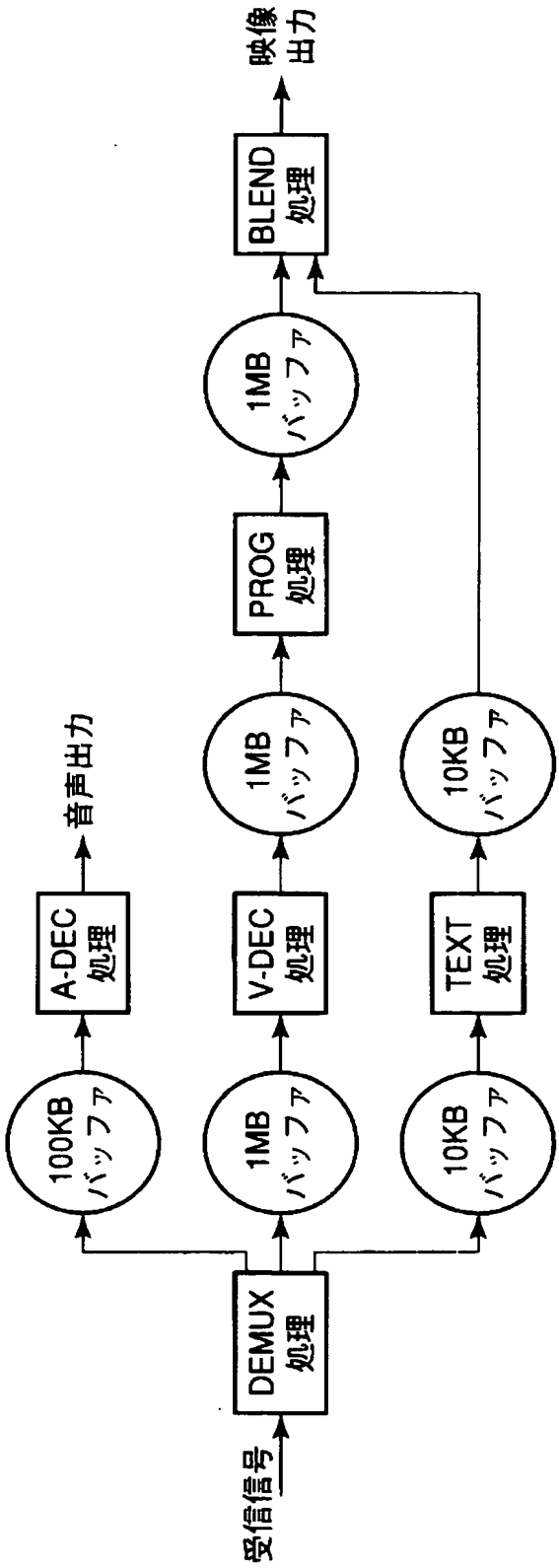

プログラムモジュール
100
DEMUXプログラム
111
A-DECプログラム
112
V-DECプログラム
113
TEXTプログラム
114
PROGプログラム
115
BLENDプログラム
116
構成記述
117

【図８】

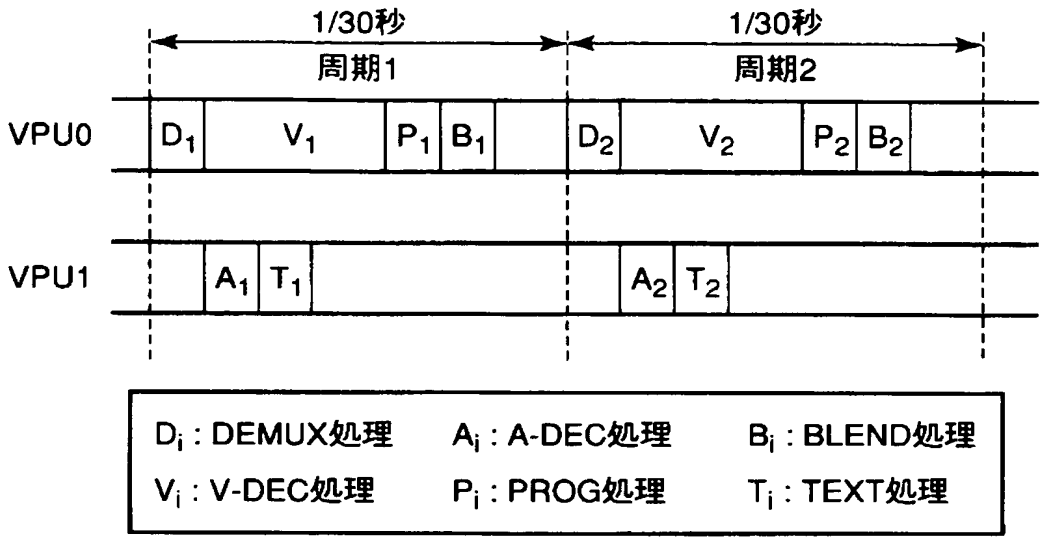
117 構成記述の例

番号	プログラム	入力	出力	コスト	バッファ
①	DEMUX	受信信号	② ③ ④	5	100KB 1MB 10KB
②	A-DEC	①	音声出力	10	———
③	V-DEC	①	⑤	50	1MB
④	TEXT	①	⑥	5	10KB
⑤	PROG	③	⑥	20	1MB
⑥	BLEND	④ ⑤	映像出力	10	———
スレッドパラメータ					
その他					

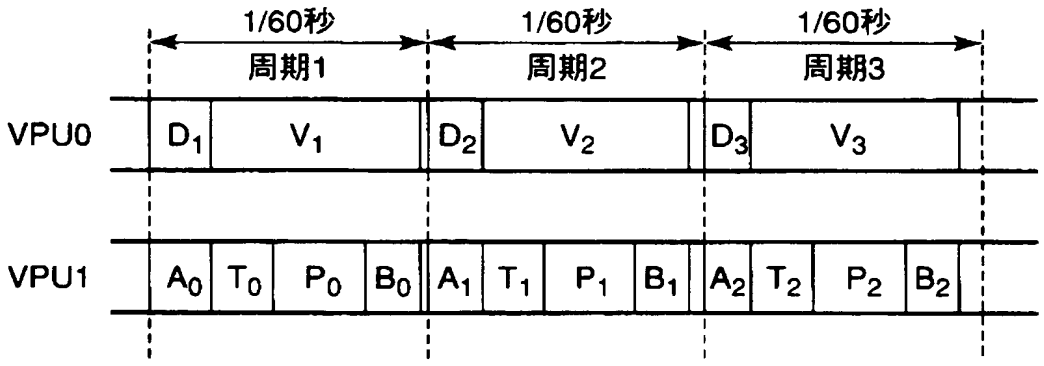
【図９】


受信信号
DEMUX 処理
100KB バッファ
1MB バッファ
10KB バッファ
A-DEC 処理
V-DEC 処理
TEXT 処理
音声出力
1MB バッファ
10KB バッファ
PROG 処理
1MB バッファ
BLEND 処理
映像出力

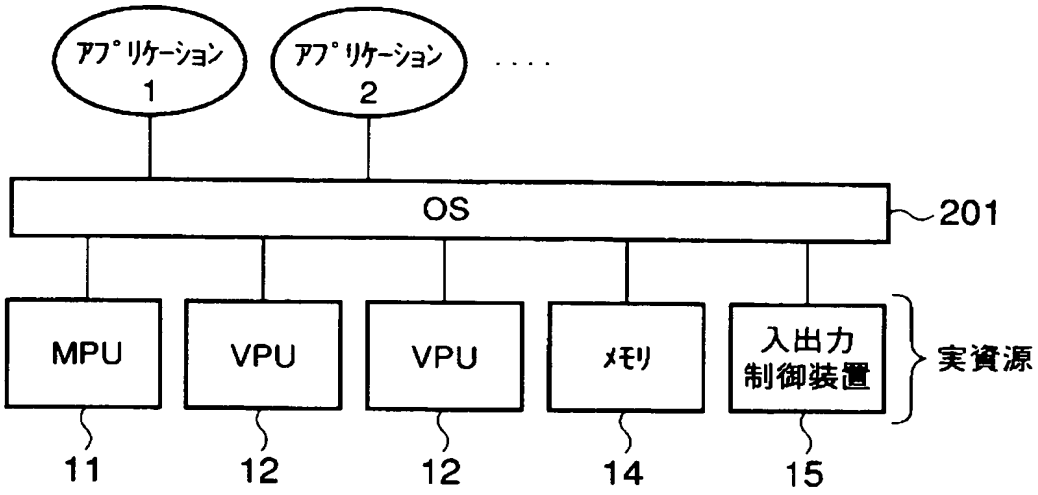
【図１０】


1/30秒
周期1
1/30秒
周期2
VPU0
D1
V1
P1
B1
D2
V2
P2
B2
VPU1
A1
T1
A2
T2
Di：DEMUX処理
Ai：A-DEC処理
Bi：BLEND処理
Vi：V-DEC処理
Pi：PROG処理
Ti：TEXT処理

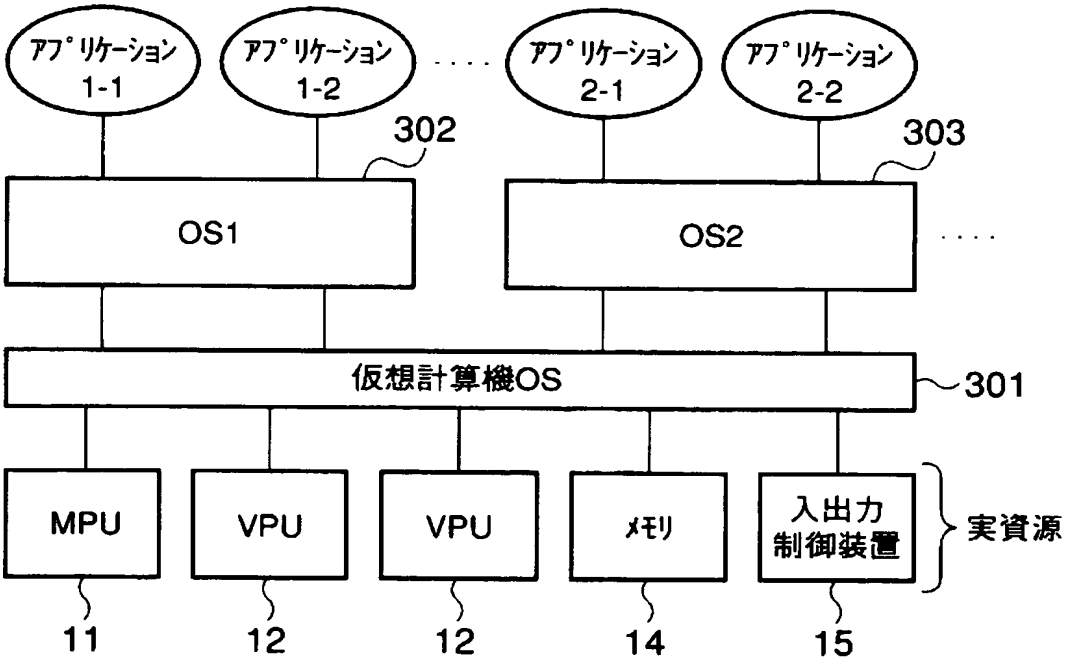
【図１１】


1/60秒
周期1
1/60秒
周期2
1/60秒
周期3
VPU0
D1
V1
D2
V2
D3
V3
VPU1
A0
T0
P0
B0
A1
T1
P1
B1
A2
T2
P2
B2

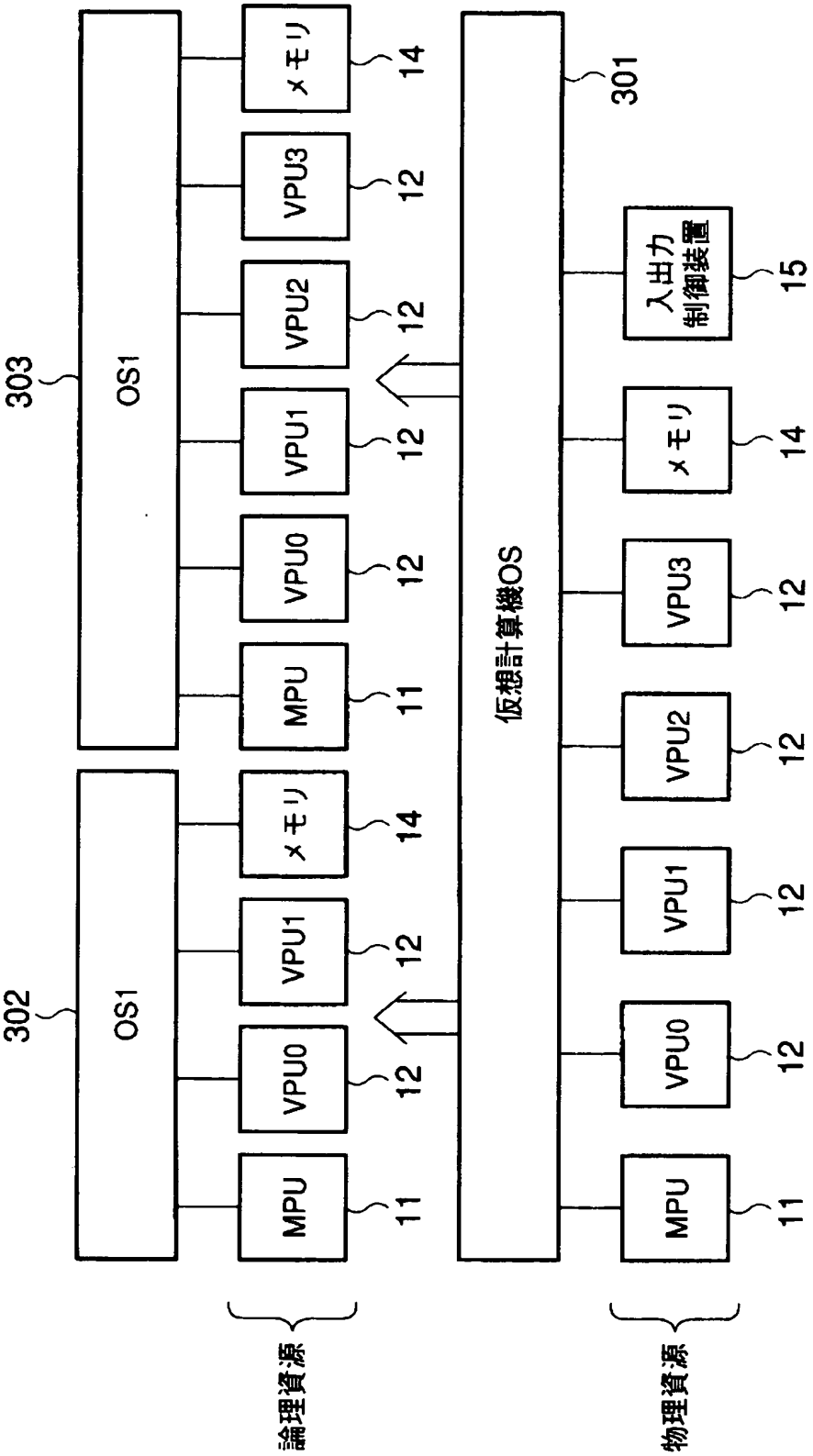
【図１２】


アプリケーション 1
アプリケーション 2
....
OS
201
MPU
VPU
VPU
メモリ
入出力制御装置
実資源
11
12
12
14
15

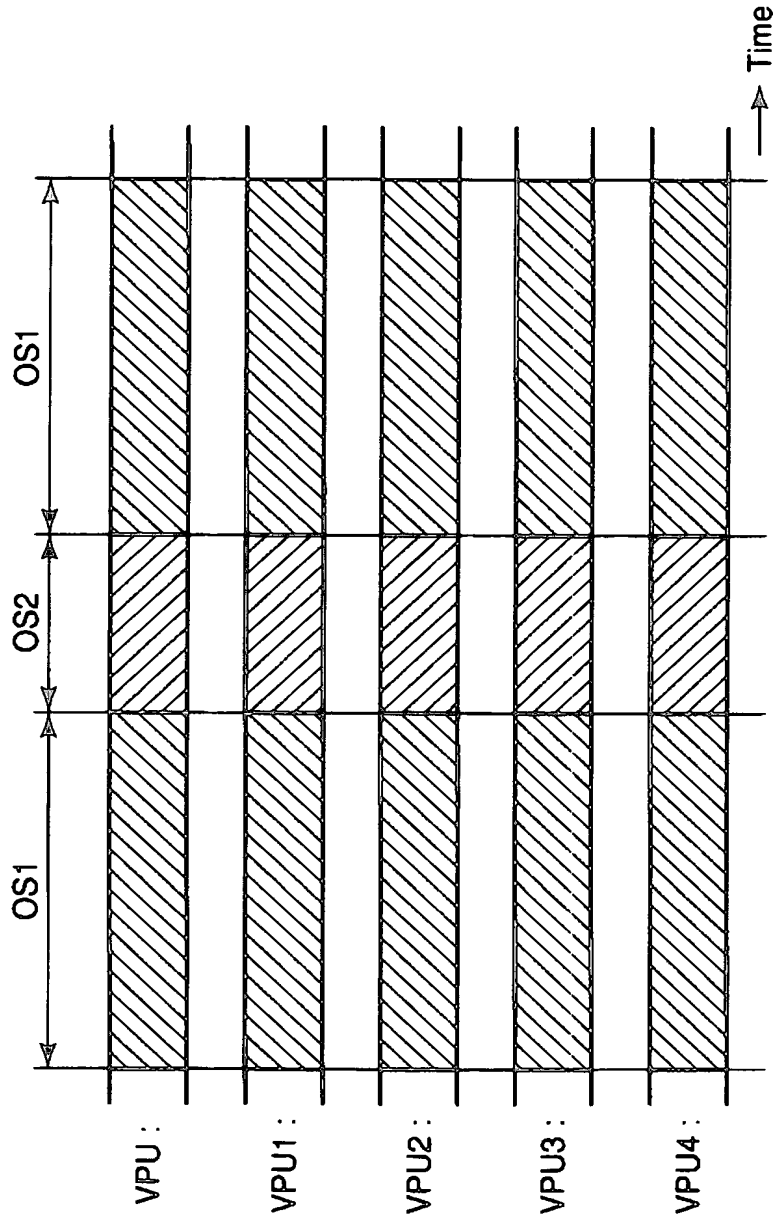
【図１３】


アプリケーション 1-1
アプリケーション 1-2
....
アプリケーション 2-1
アプリケーション 2-2
302
303
OS1
OS2
....
仮想計算機OS
301
MPU
VPU
VPU
メモリ
入出力制御装置
実資源
11
12
12
14
15

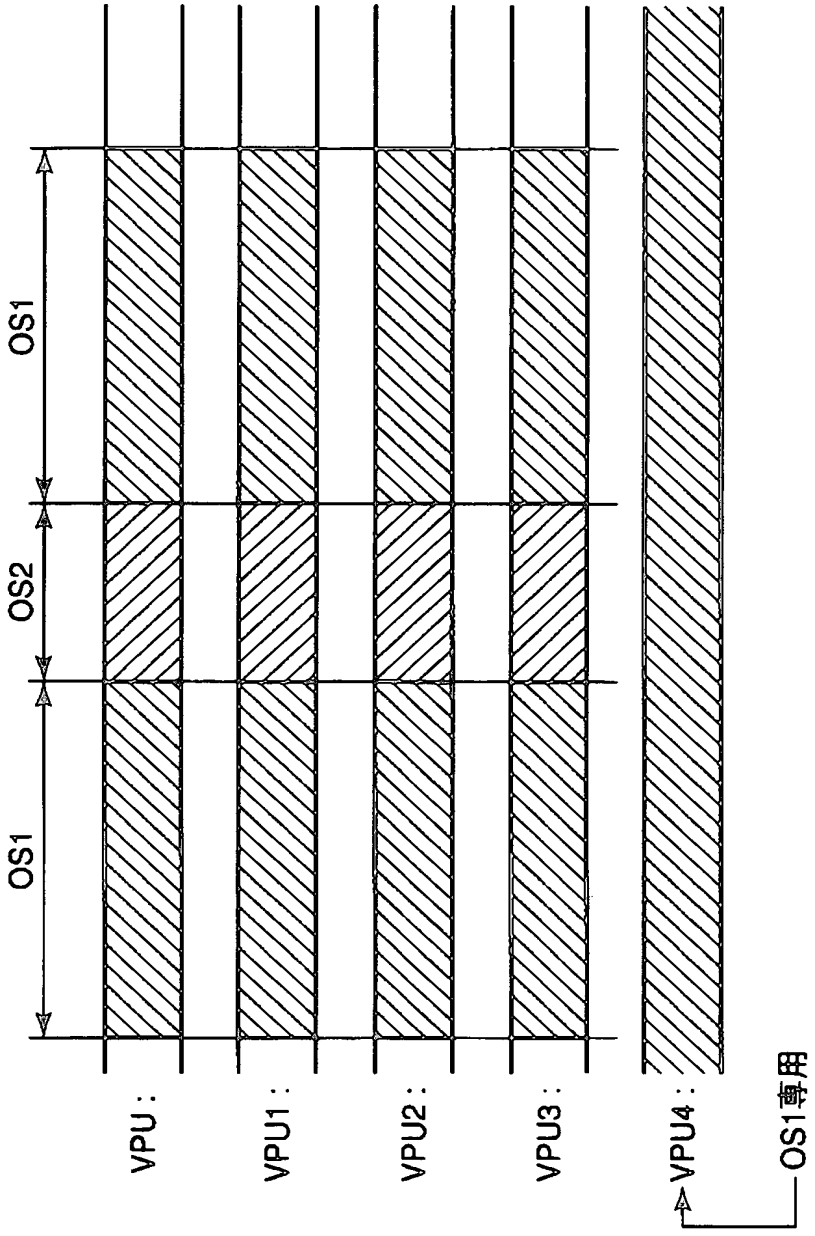
【図１４】


303
OS1
302
OS1
MPU
VPU0
VPU1
メモリ
VPU2
VPU3
11
12
14
仮想計算機OS
301
入出力制御装置
15
論理資源
物理資源

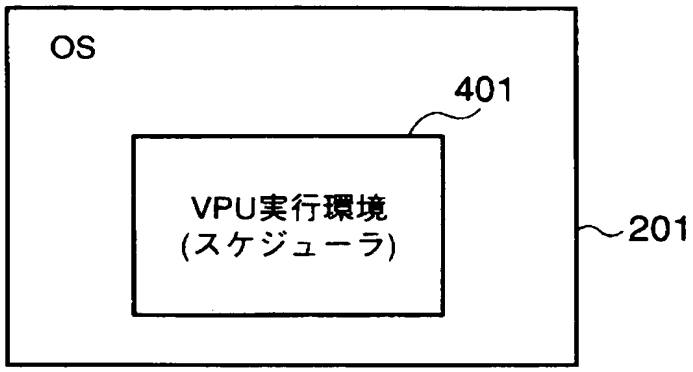
【図１５】



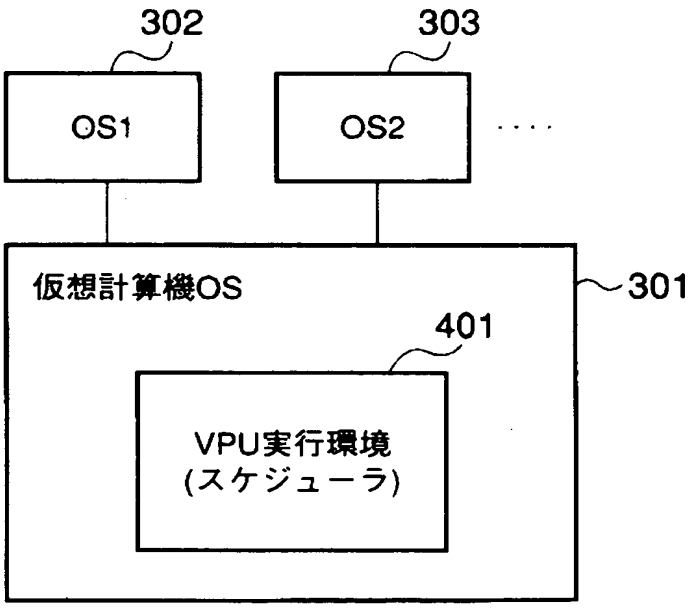
【図１６】



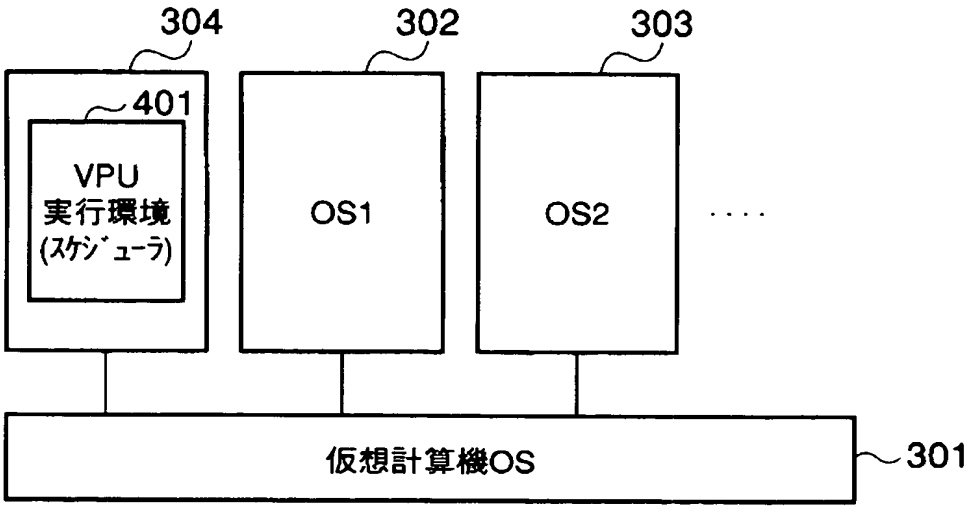
【図１７】


OS
401
VPU実行環境
(スケジューラ)
201

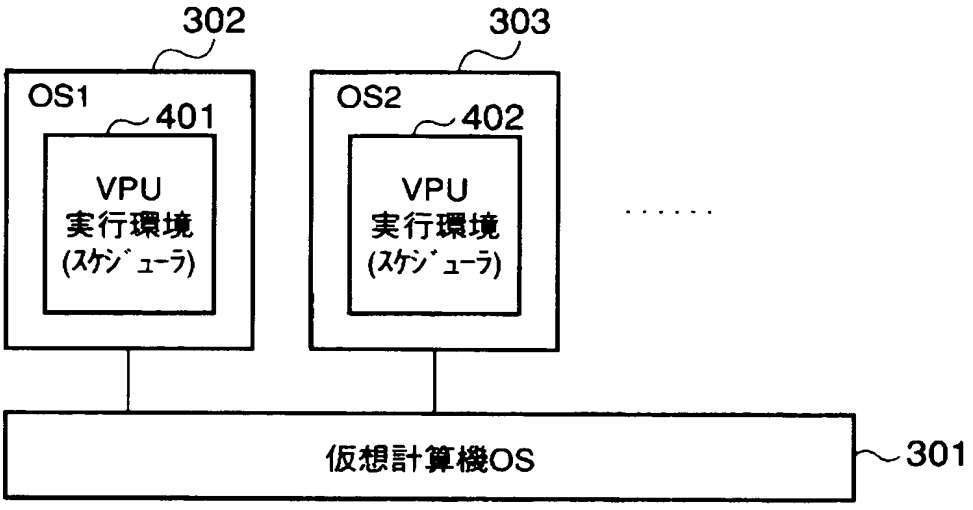
【図１８】


302
303
OS1
OS2
仮想計算機OS
301
401
VPU実行環境
(スケジューラ)

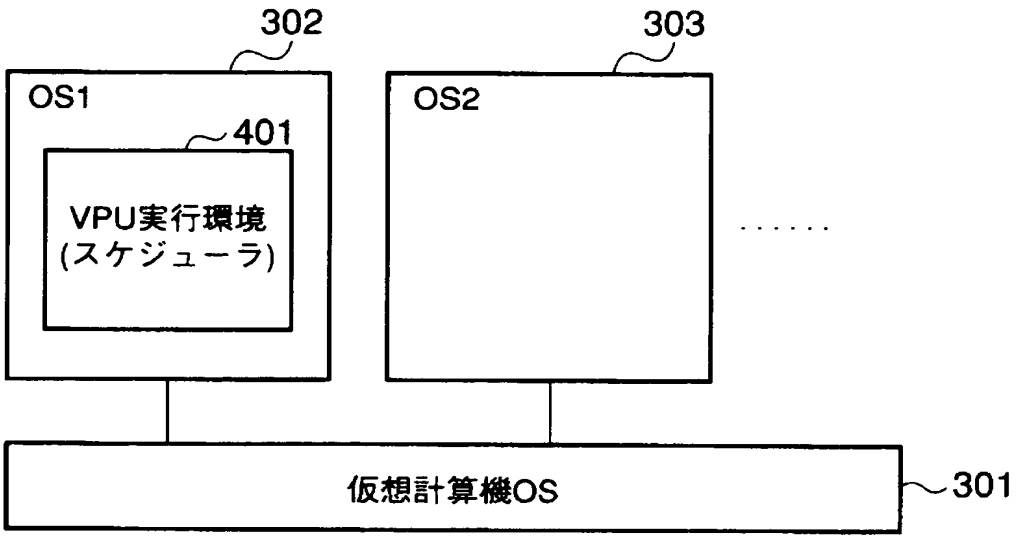
【図１９】


304
302
303
401
VPU
実行環境
(スケジューラ)
OS1
OS2
仮想計算機OS
301

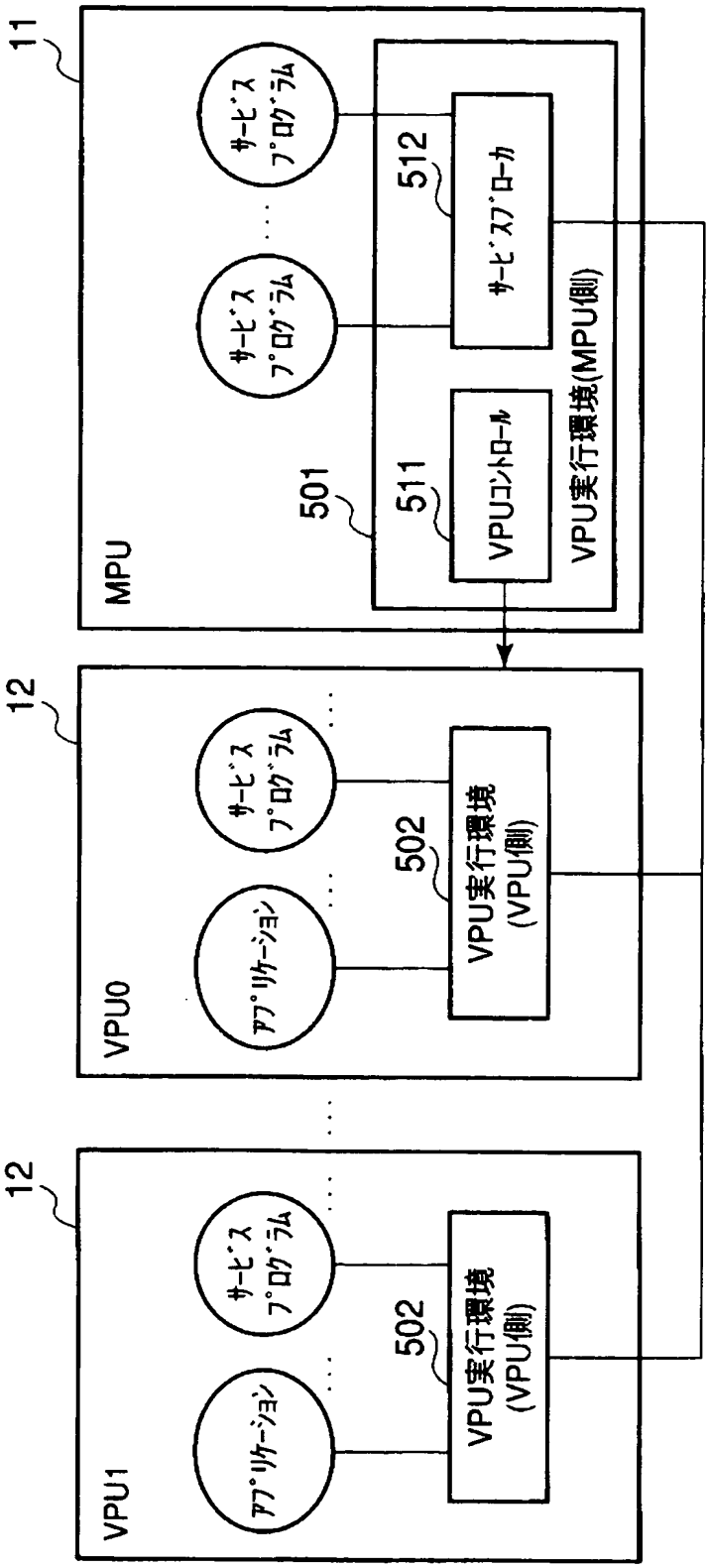
【図２０】


302
303
OS1
401
VPU
実行環境
(スケジューラ)
OS2
402
VPU
実行環境
(スケジューラ)
仮想計算機OS
301

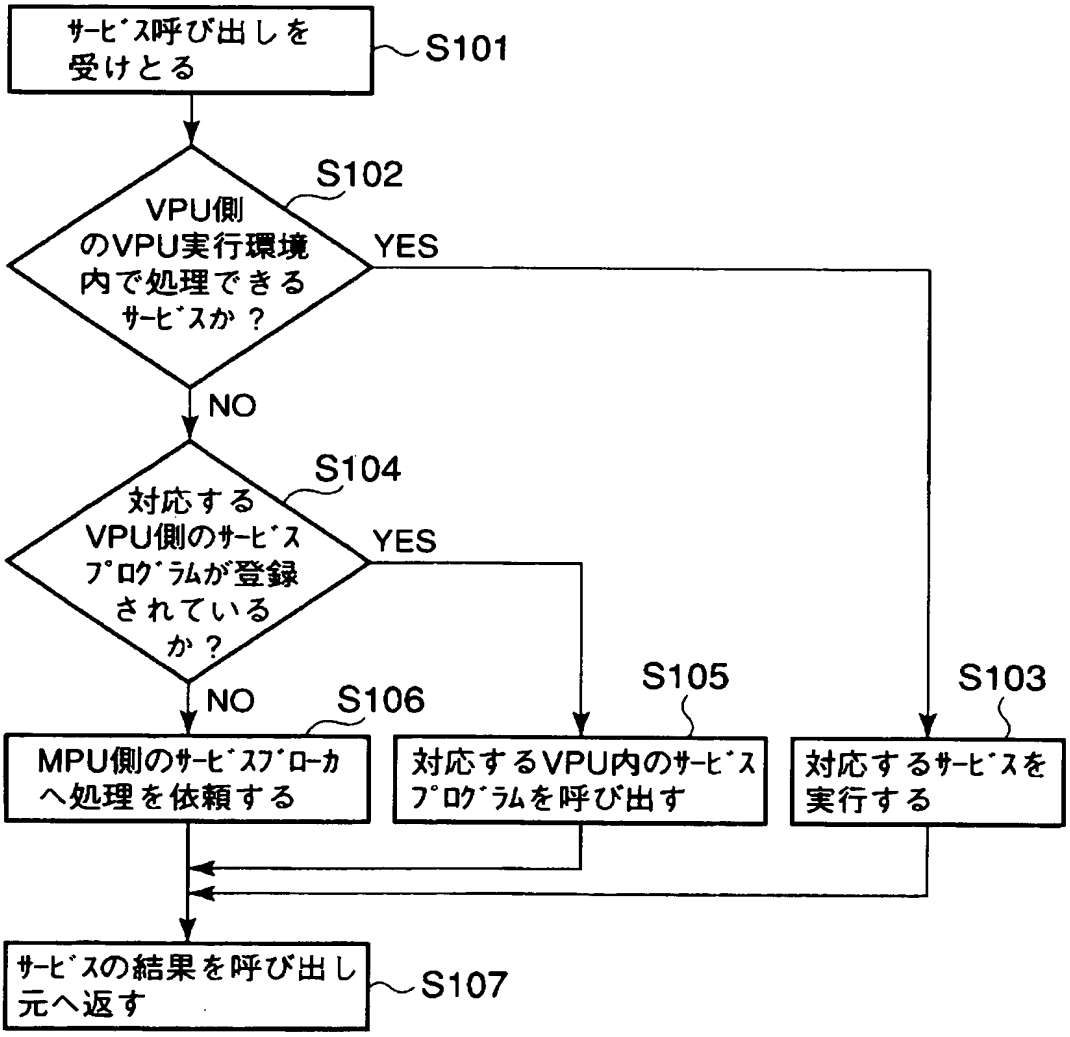
【図２１】



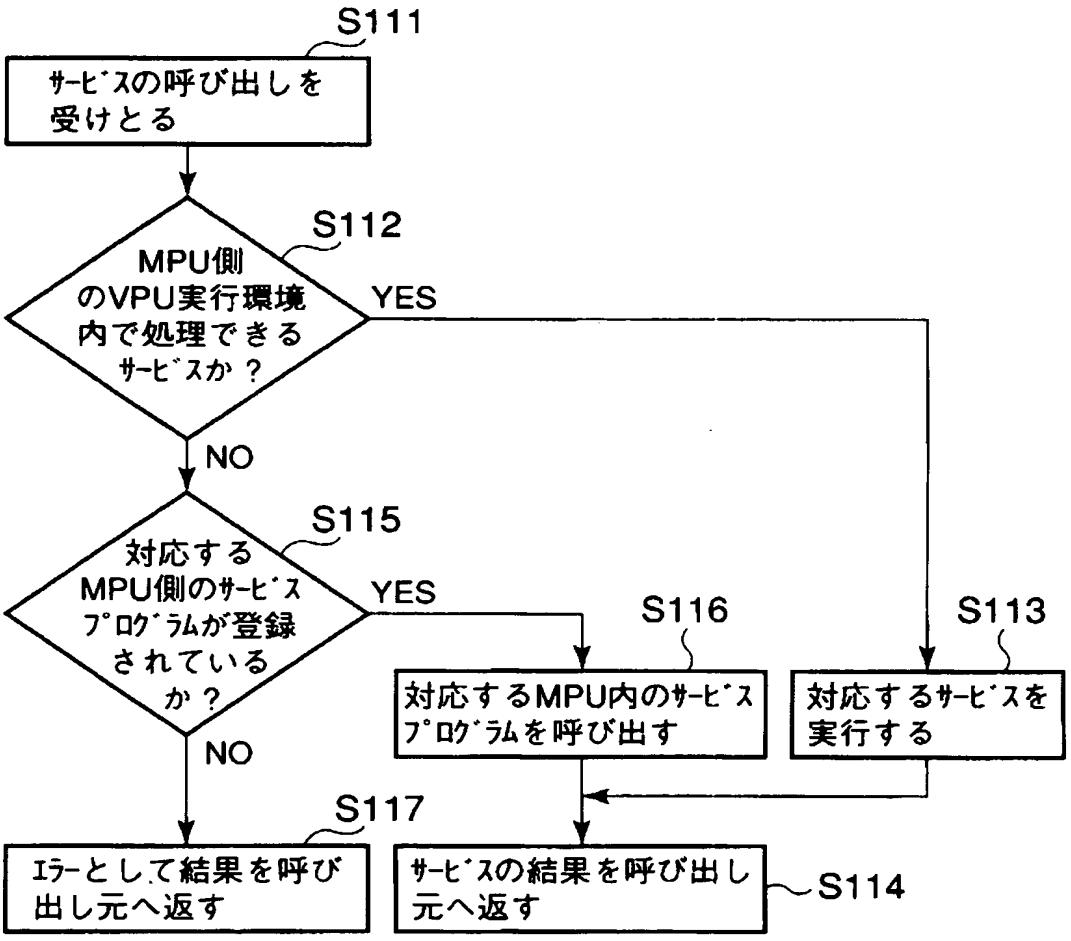
【図２２】


11
MPU
サービス プログラム
サービス プログラム
512
サービスブローカ
501
511
VPUコントロール
VPU実行環境(MPU側)
12
VPU0
サービス プログラム
アプリケーション
502
VPU実行環境 (VPU側)
12
VPU1
サービス プログラム
アプリケーション
502
VPU実行環境 (VPU側)

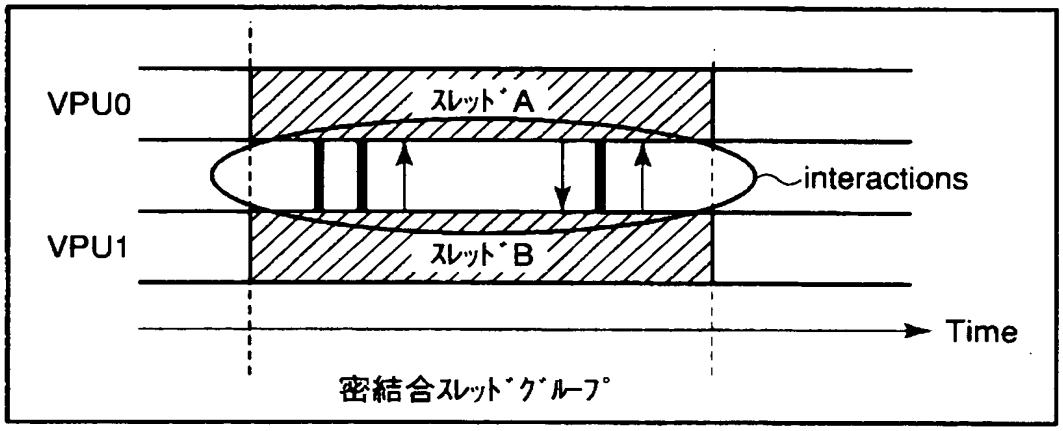
【図２３】


ｻｰﾋﾞｽ呼び出しを受けとる
S101
S102
VPU側のVPU実行環境内で処理できるｻｰﾋﾞｽか？
YES
NO
S104
対応するVPU側のｻｰﾋﾞｽﾌﾟﾛｸﾞﾗﾑが登録されているか？
YES
NO
S106
S105
S103
MPU側のｻｰﾋﾞｽﾌﾞﾛｰｶへ処理を依頼する
対応するVPU内のｻｰﾋﾞｽﾌﾟﾛｸﾞﾗﾑを呼び出す
対応するｻｰﾋﾞｽを実行する
ｻｰﾋﾞｽの結果を呼び出し元へ返す
S107

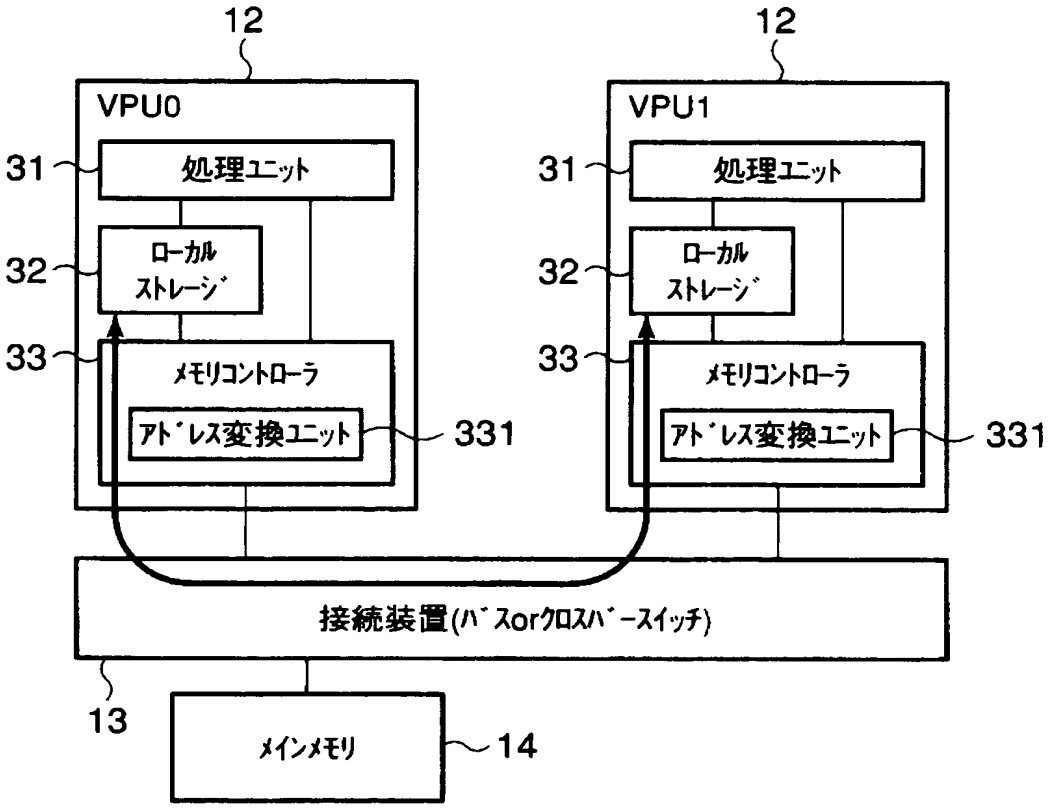
【図２４】


S111
ｻｰﾋﾞｽの呼び出しを受けとる
S112
MPU側のVPU実行環境内で処理できるｻｰﾋﾞｽか？
YES
NO
S115
対応するMPU側のｻｰﾋﾞｽﾌﾟﾛｸﾞﾗﾑが登録されているか？
YES
NO
S116
対応するMPU内のｻｰﾋﾞｽﾌﾟﾛｸﾞﾗﾑを呼び出す
S113
対応するｻｰﾋﾞｽを実行する
S117
ｴﾗｰとして結果を呼び出し元へ返す
ｻｰﾋﾞｽの結果を呼び出し元へ返す
S114

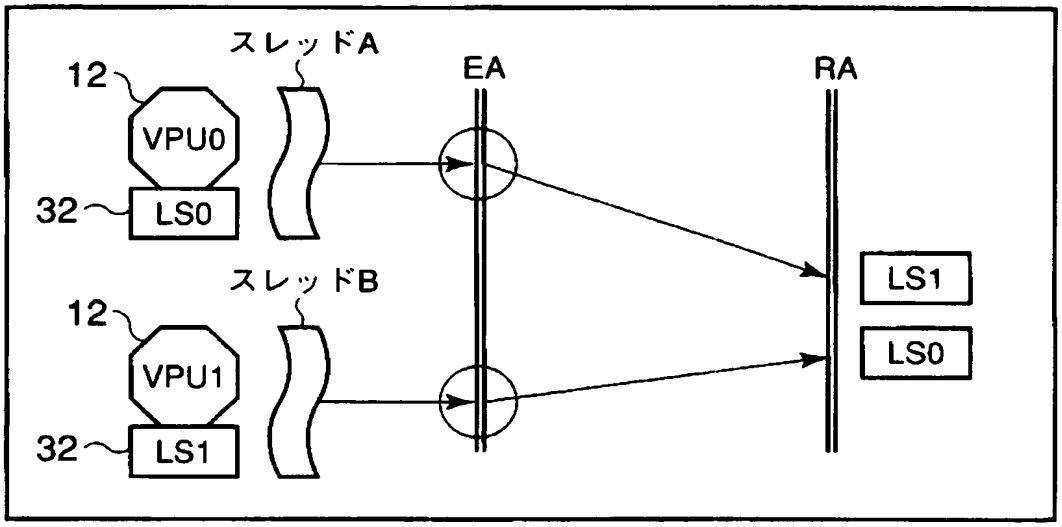
【図２５】


VPU0
ｽﾚｯﾄﾞA
interactions
VPU1
ｽﾚｯﾄﾞB
Time
密結合ｽﾚｯﾄﾞｸﾞﾙｰﾌﾟ

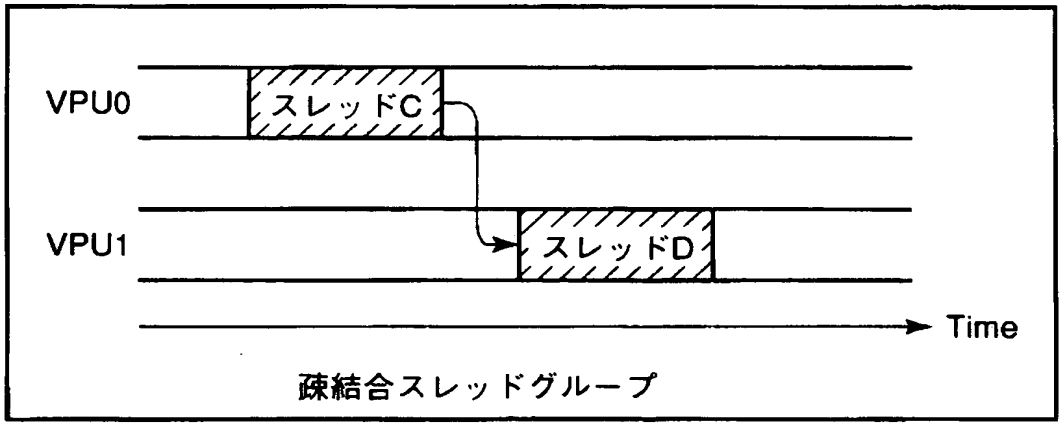
【図２６】


12
12
VPU0
VPU1
31
処理ユニット
32
ローカル
ストレージ
33
メモリコントローラ
アドレス変換ユニット
331
接続装置(バスorクロスバースイッチ)
13
メインメモリ
14

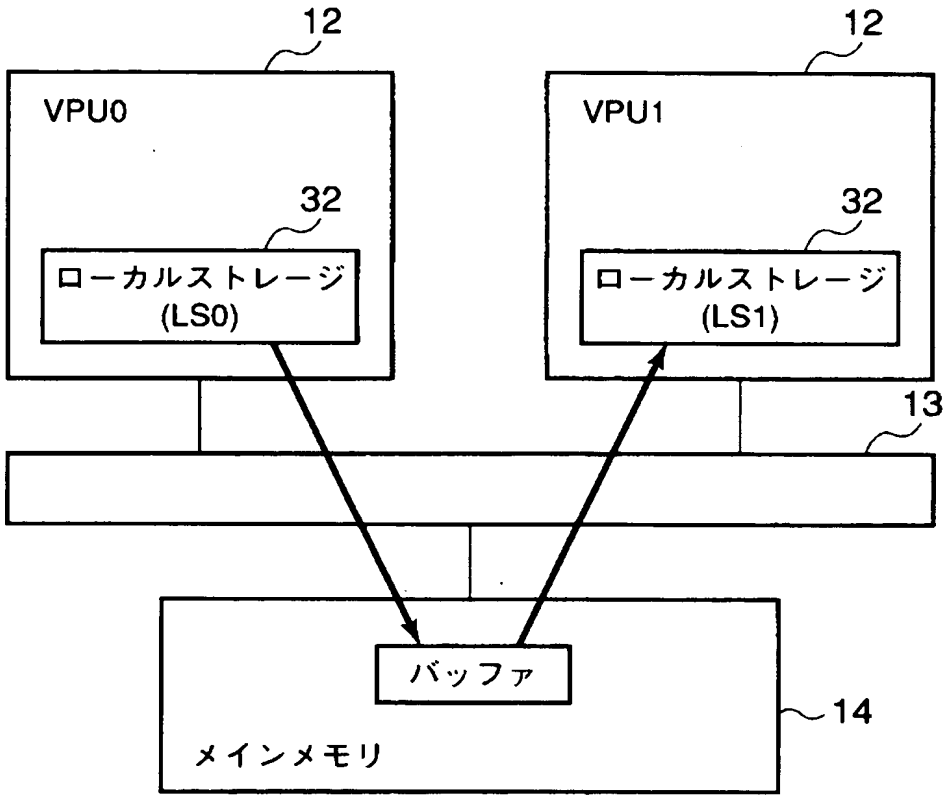
【図２７】


スレッドA
EA
RA
12
VPU0
32
LS0
スレッドB
12
VPU1
32
LS1
LS1
LS0

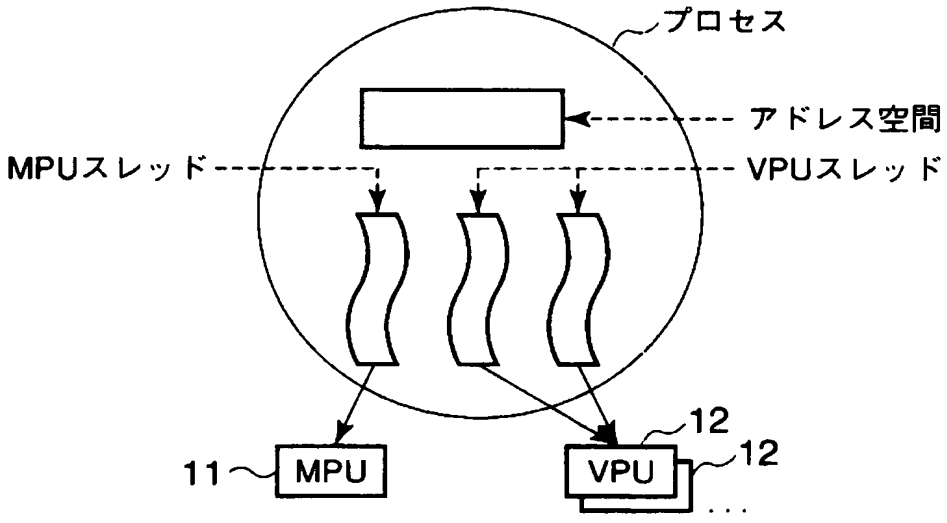
【図２８】


VPU0
スレッドC
VPU1
スレッドD
Time
疎結合スレッドグループ

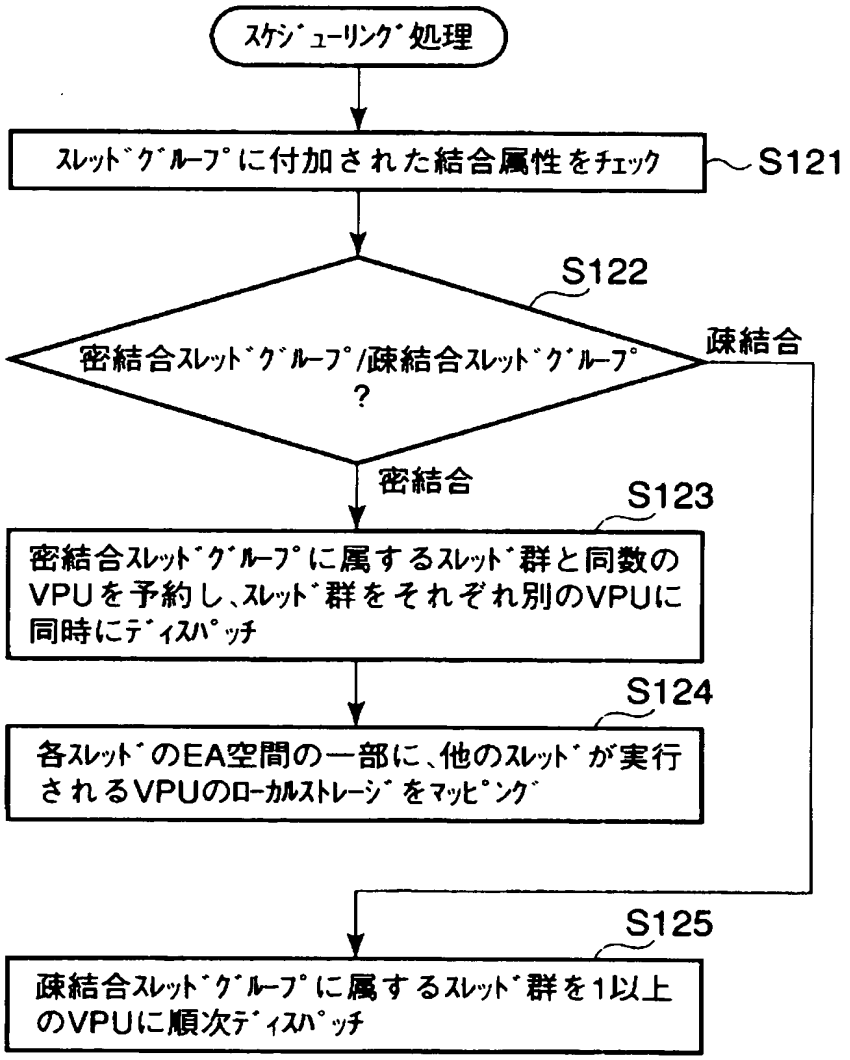
【図２９】


12
12
VPU0
VPU1
32
32
ローカルストレージ
(LS0)
ローカルストレージ
(LS1)
13
バッファ
14
メインメモリ

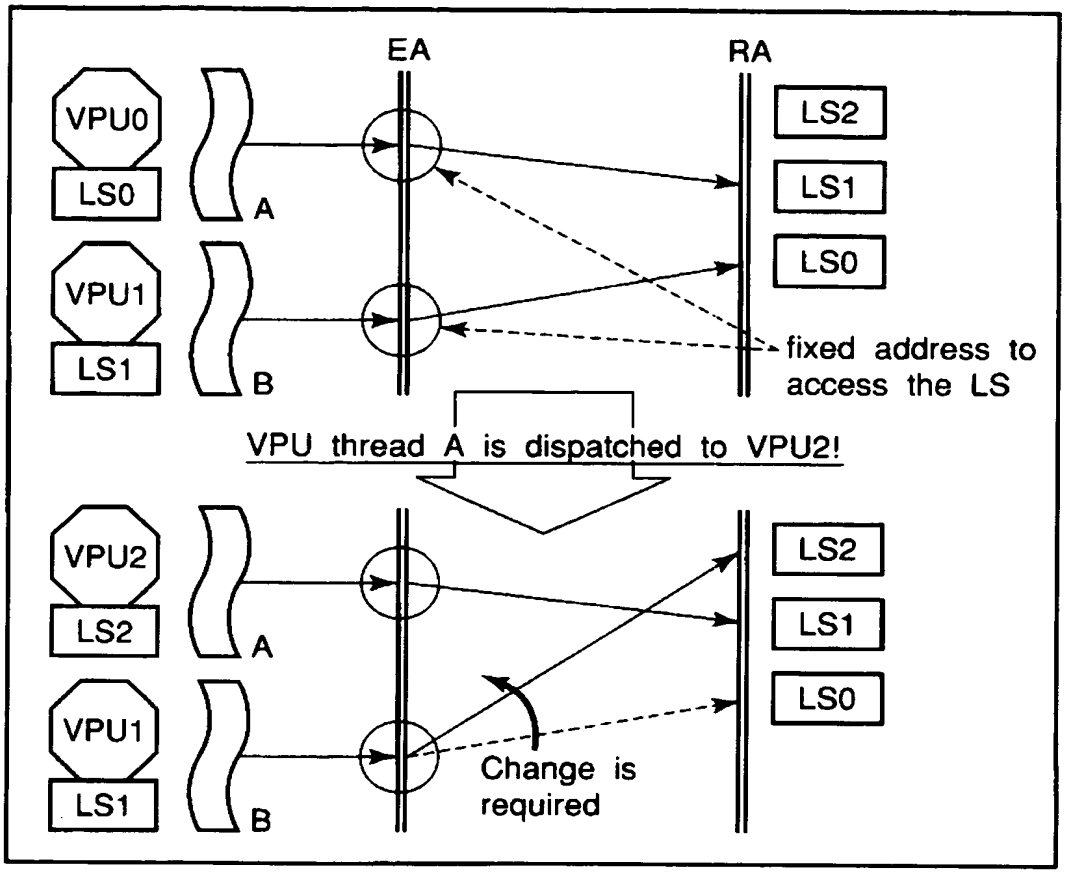
【図３０】


プロセス
アドレス空間
MPUスレッド
VPUスレッド
12
11
MPU
VPU
12

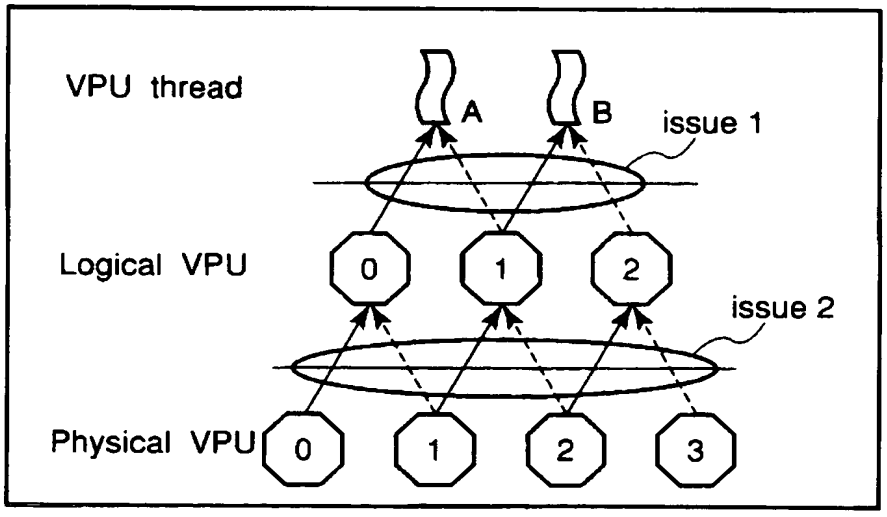
【図３１】


ｽｹｼﾞｭｰﾘﾝｸﾞ処理
ｽﾚｯﾄﾞｸﾞﾙｰﾌﾟに付加された結合属性をﾁｪｯｸ
S121
S122
密結合ｽﾚｯﾄﾞｸﾞﾙｰﾌﾟ/疎結合ｽﾚｯﾄﾞｸﾞﾙｰﾌﾟ
?
疎結合
密結合
S123
密結合ｽﾚｯﾄﾞｸﾞﾙｰﾌﾟに属するｽﾚｯﾄﾞ群と同数のVPUを予約し、ｽﾚｯﾄﾞ群をそれぞれ別のVPUに同時にﾃﾞｨｽﾊﾟｯﾁ
S124
各ｽﾚｯﾄﾞのEA空間の一部に、他のｽﾚｯﾄﾞが実行されるVPUのﾛｰｶﾙｽﾄﾚｰｼﾞをﾏｯﾋﾟﾝｸﾞ
S125
疎結合ｽﾚｯﾄﾞｸﾞﾙｰﾌﾟに属するｽﾚｯﾄﾞ群を1以上のVPUに順次ﾃﾞｨｽﾊﾟｯﾁ

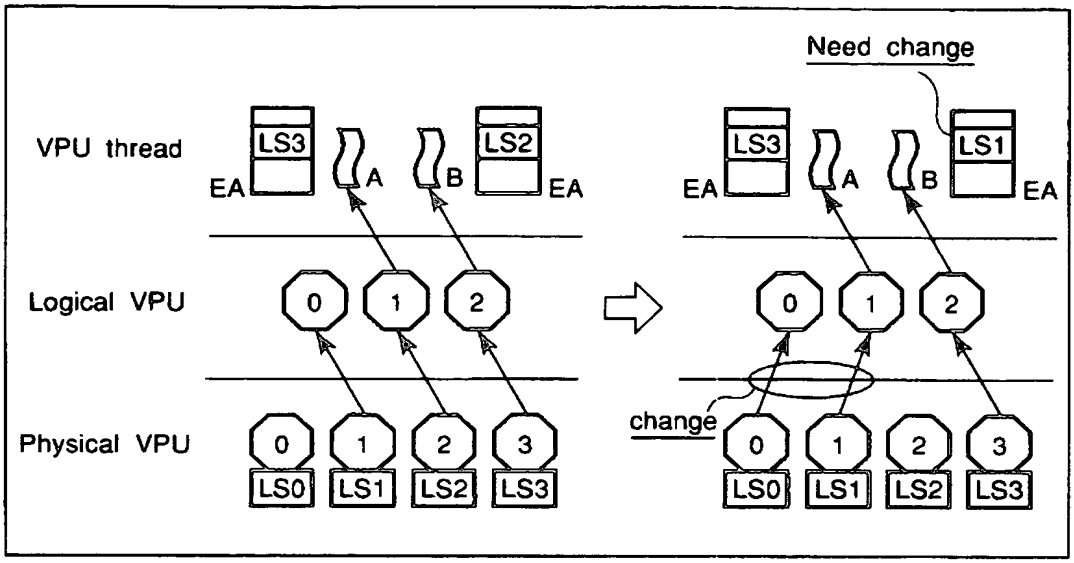
【図３２】


EA
RA
VPU0
LS0
A
VPU1
LS1
B
LS2
LS1
LS0
fixed address to access the LS
VPU thread A is dispatched to VPU2!
VPU2
LS2
A
VPU1
LS1
B
LS2
LS1
LS0
Change is required

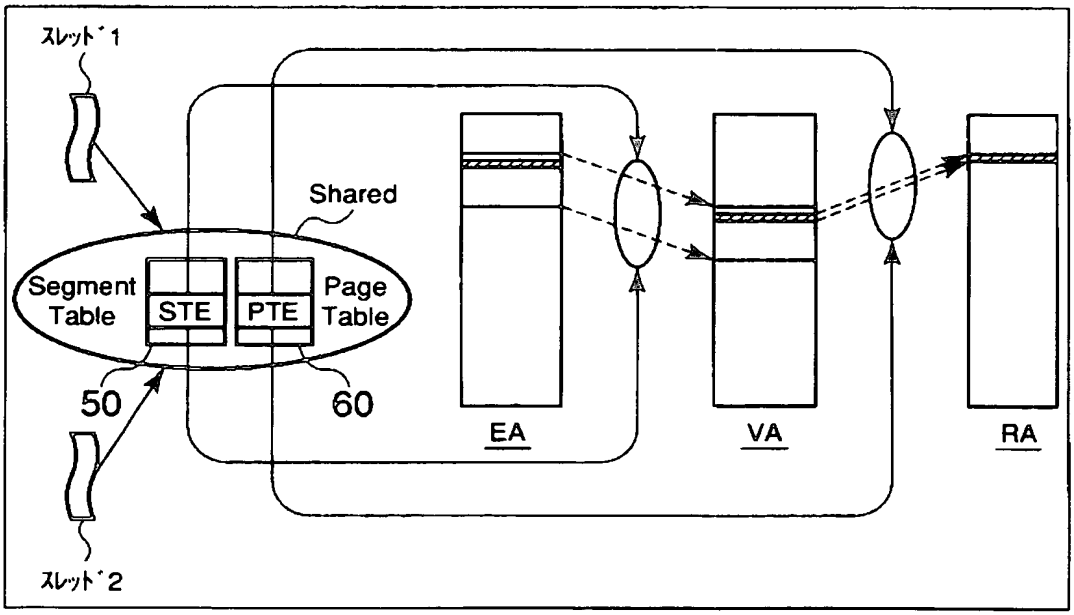
【図３３】


VPU thread
A
B
issue 1
Logical VPU
0
1
2
issue 2
Physical VPU
0
1
2
3

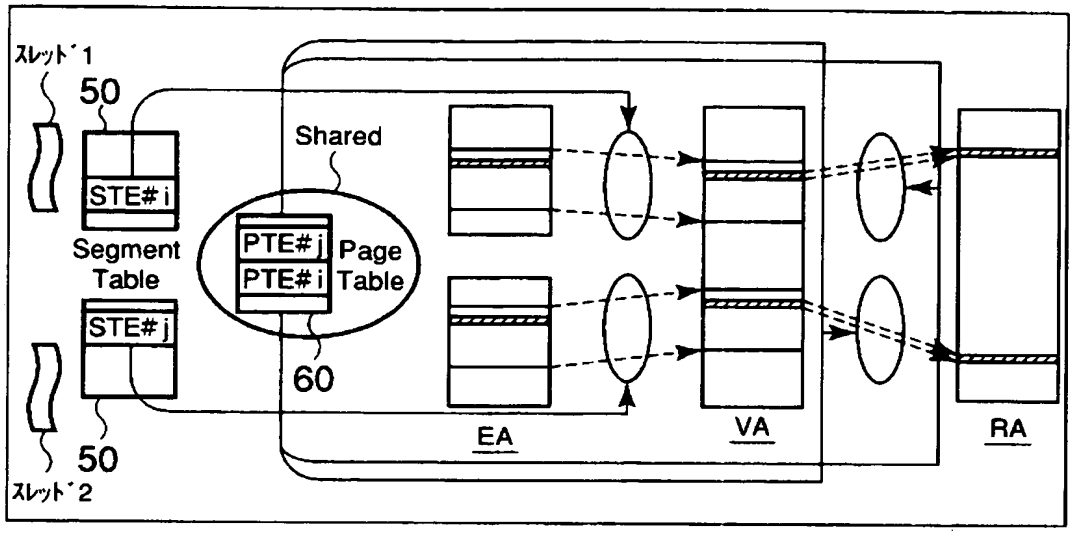
【図３４】


Need change
VPU thread
LS3
A
B
LS2
EA
EA
LS3
A
B
LS1
EA
Logical VPU
0
1
2
0
1
2
Physical VPU
change
0
1
2
3
LS0
LS1
LS2
LS3
0
1
2
3
LS0
LS1
LS2
LS3

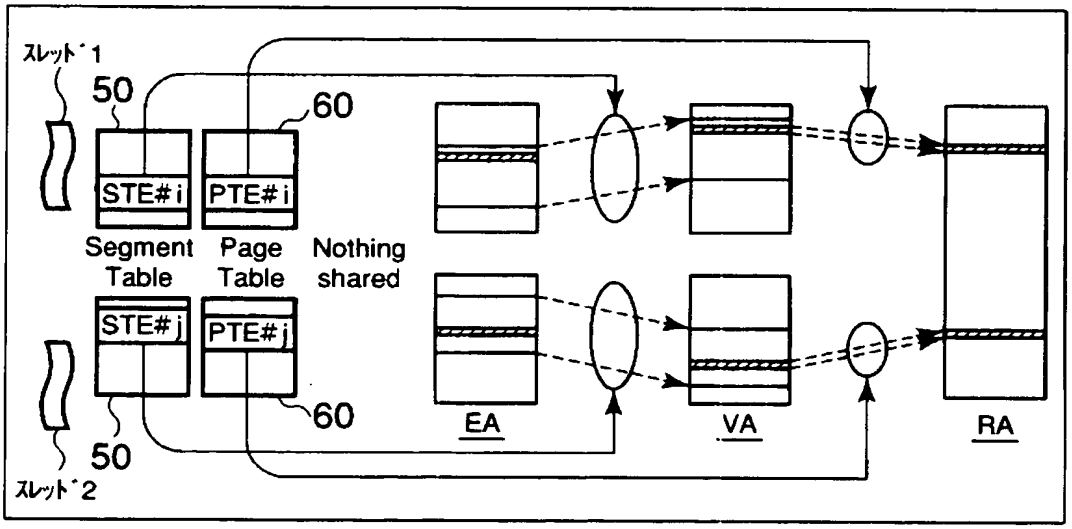
【図３５】


スレッド1
Shared
Segment Table
STE
PTE
Page Table
50
60
EA
VA
RA
スレッド2

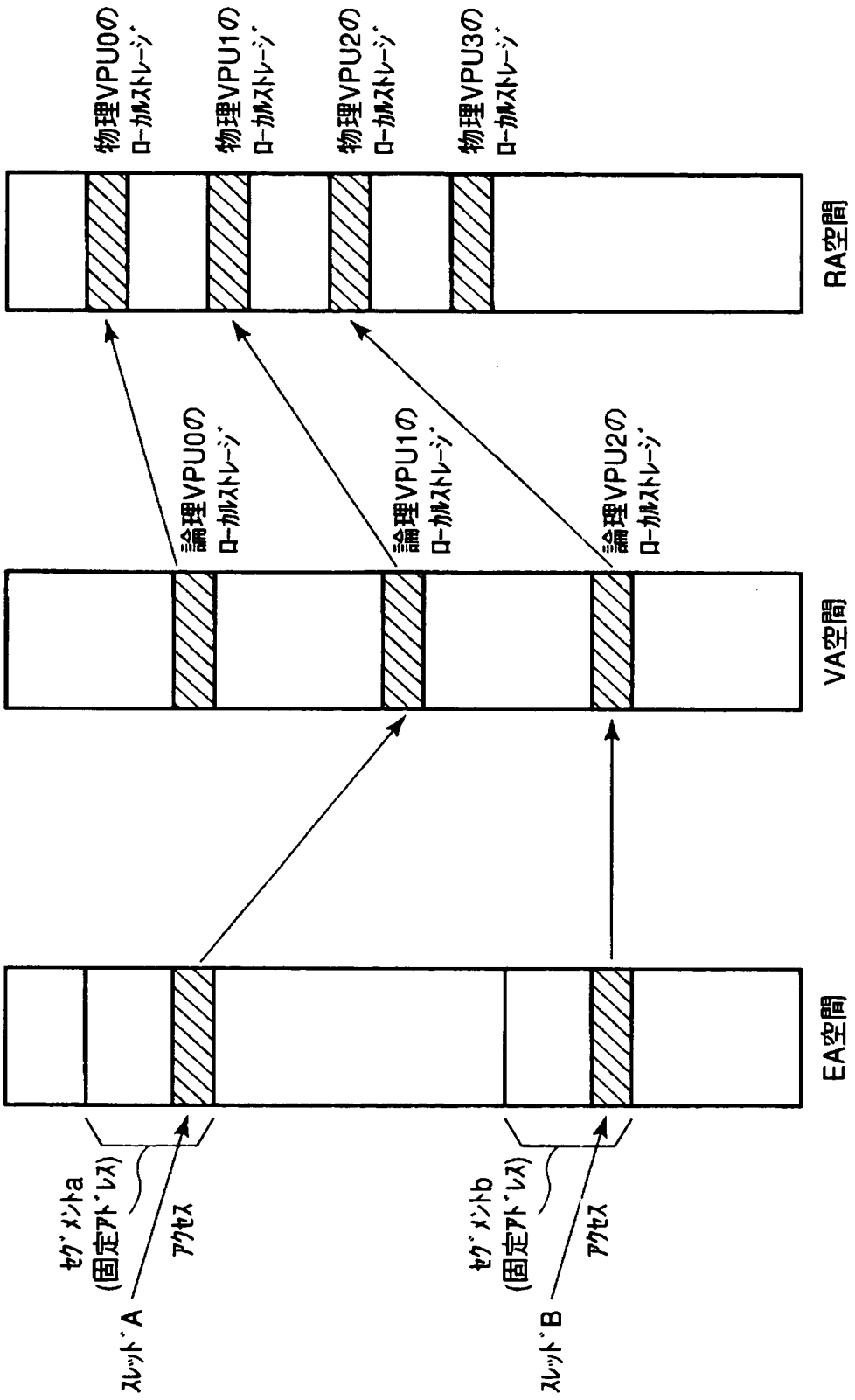
【図３６】


スレッド1
50
STE#i
Segment
Table
STE#j
50
スレッド2
Shared
PTE#j
PTE#i
Page
Table
60
EA
VA
RA

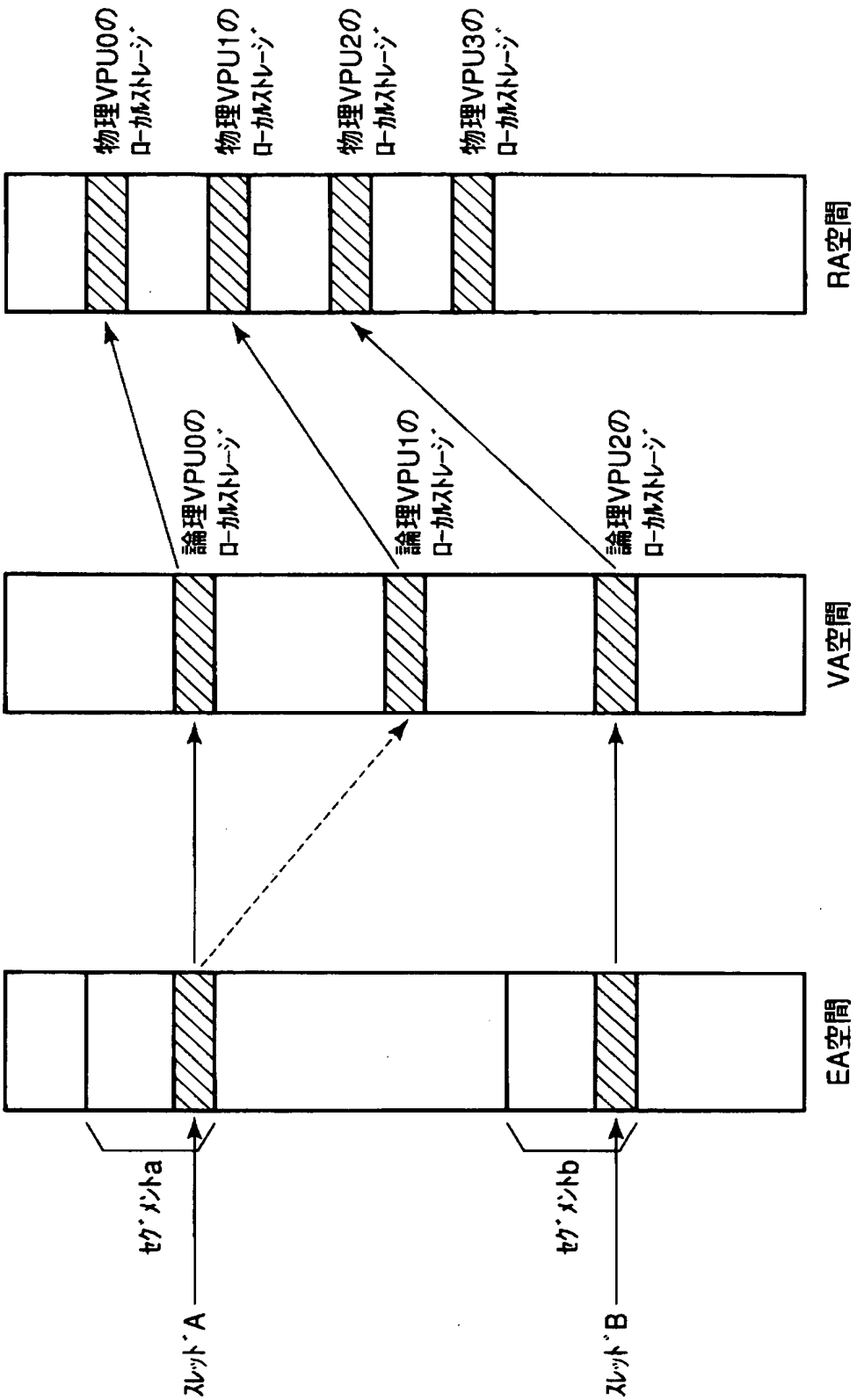
【図３７】


スレッド1
50
STE#i
PTE#i
60
Segment
Table
Page
Table
Nothing
shared
STE#j
PTE#j
60
50
スレッド2
EA
VA
RA

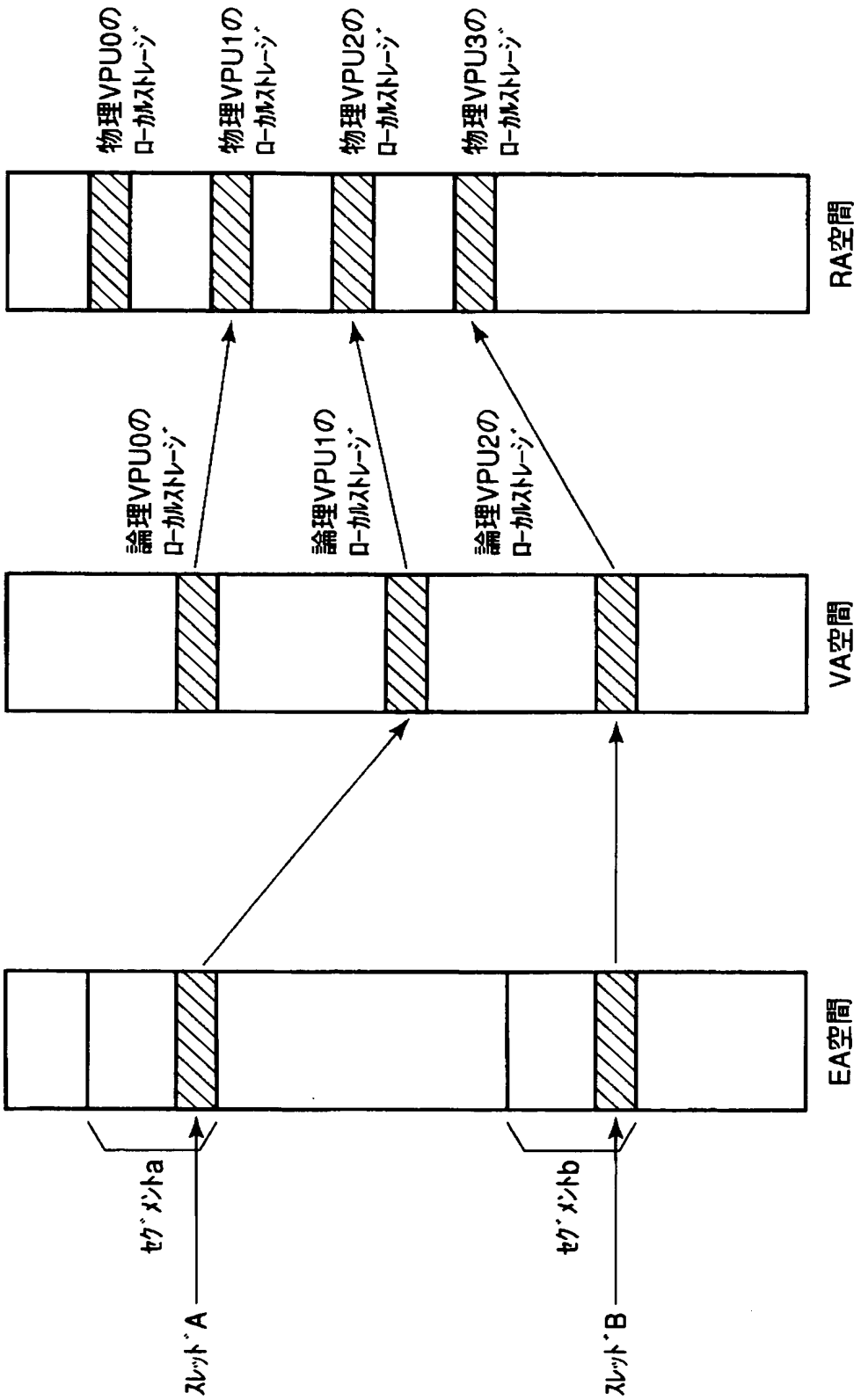
【図３８】


物理VPU0のローカルストレージ
物理VPU1のローカルストレージ
物理VPU2のローカルストレージ
物理VPU3のローカルストレージ
RA空間
論理VPU0のローカルストレージ
論理VPU1のローカルストレージ
論理VPU2のローカルストレージ
VA空間
EA空間
セグメントa
(固定アドレス)
アクセス
スレッドA
セグメントb
(固定アドレス)
アクセス
スレッドB

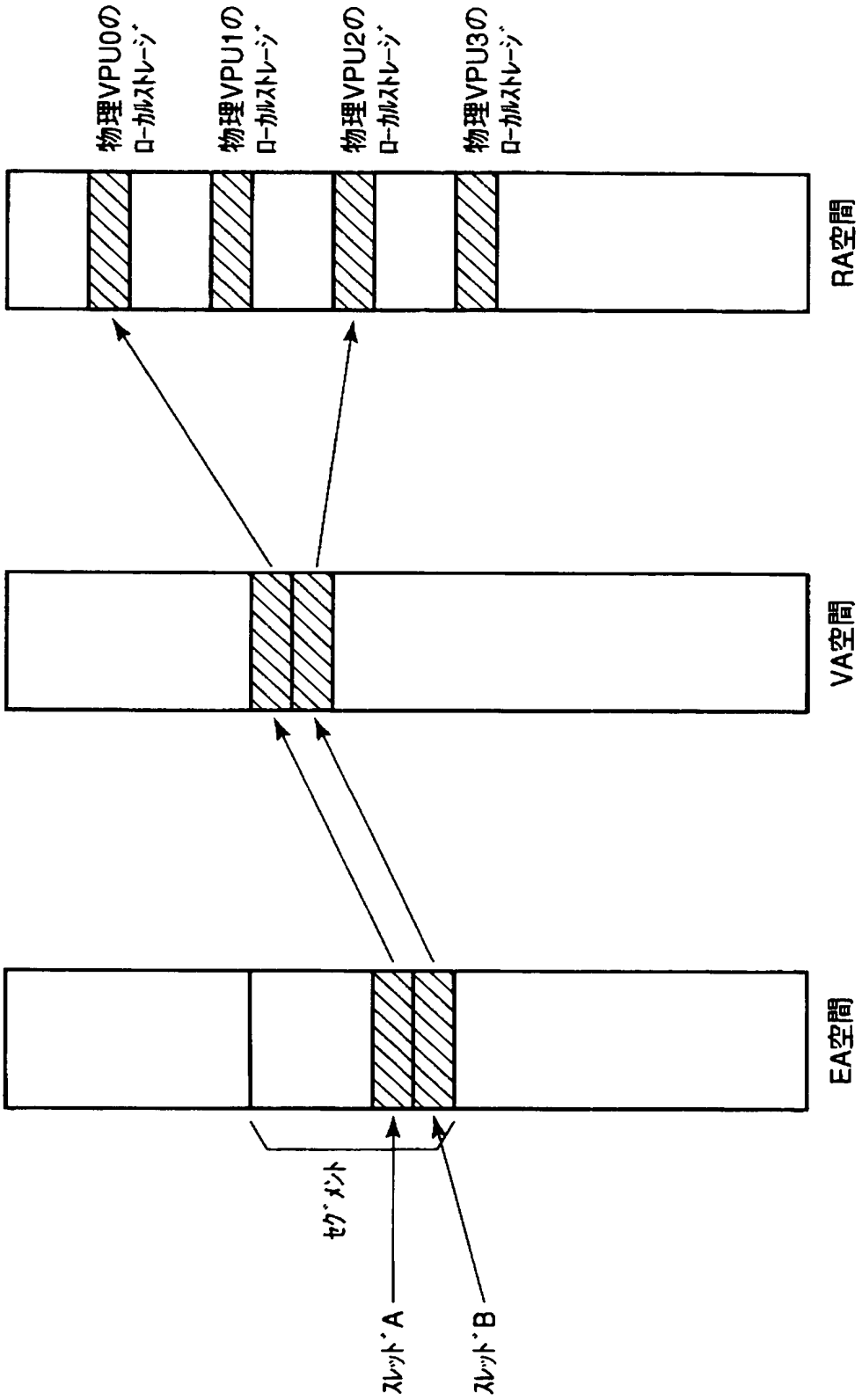
【図３９】


物理VPU0のローカルストレージ
物理VPU1のローカルストレージ
物理VPU2のローカルストレージ
物理VPU3のローカルストレージ
RA空間
論理VPU0のローカルストレージ
論理VPU1のローカルストレージ
論理VPU2のローカルストレージ
VA空間
EA空間
セグメントa
セグメントb
スレッドA
スレッドB

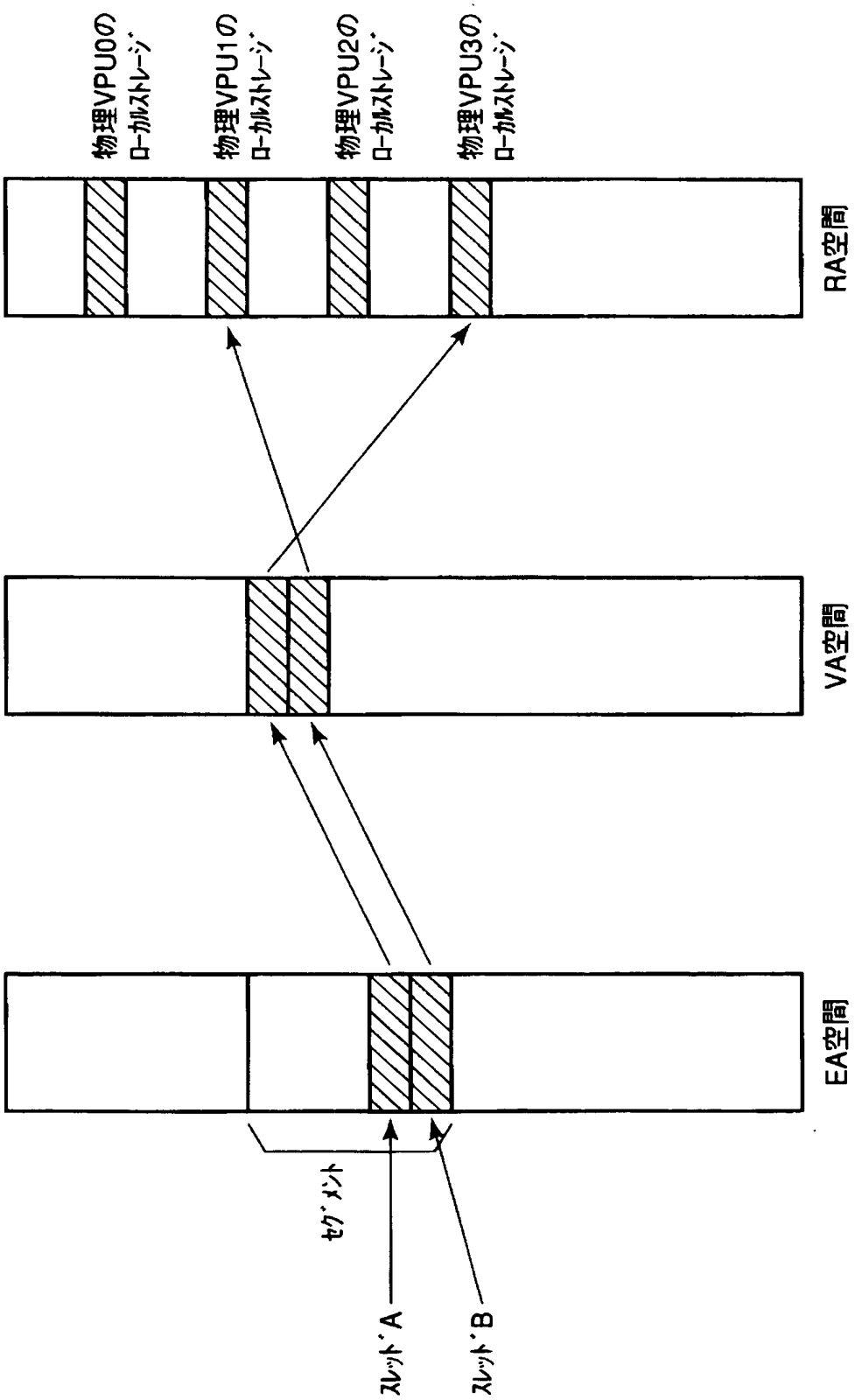
【図４０】


物理VPU0のローカルストレージ
物理VPU1のローカルストレージ
物理VPU2のローカルストレージ
物理VPU3のローカルストレージ
RA空間
論理VPU0のローカルストレージ
論理VPU1のローカルストレージ
論理VPU2のローカルストレージ
VA空間
EA空間
セグメントa
セグメントb
スレッドA
スレッドB

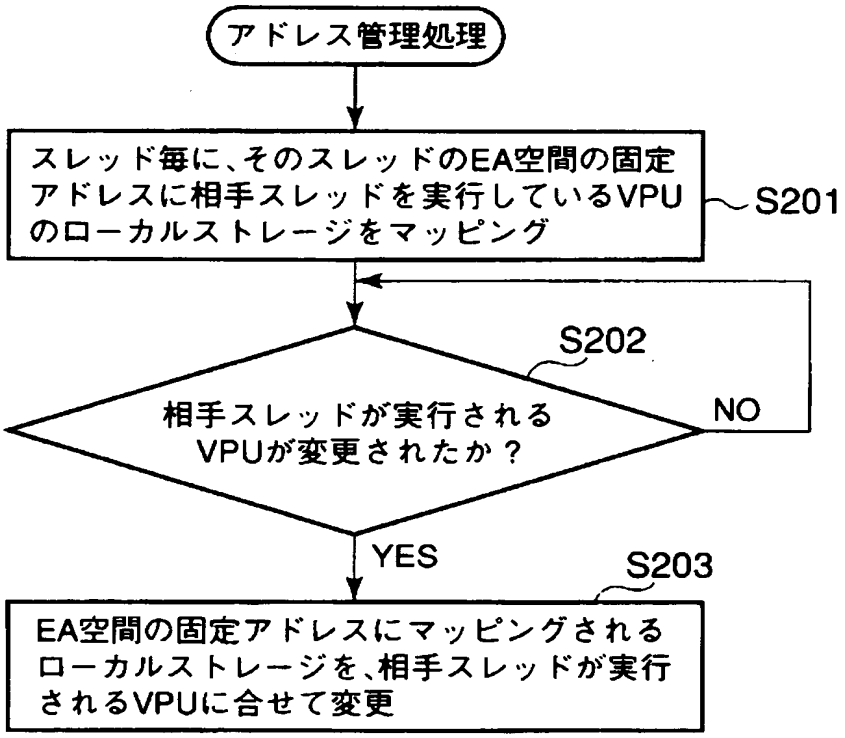
【図４１】


物理VPU0のローカルストレージ
物理VPU1のローカルストレージ
物理VPU2のローカルストレージ
物理VPU3のローカルストレージ
RA空間
VA空間
EA空間
セグメント
スレッドA
スレッドB

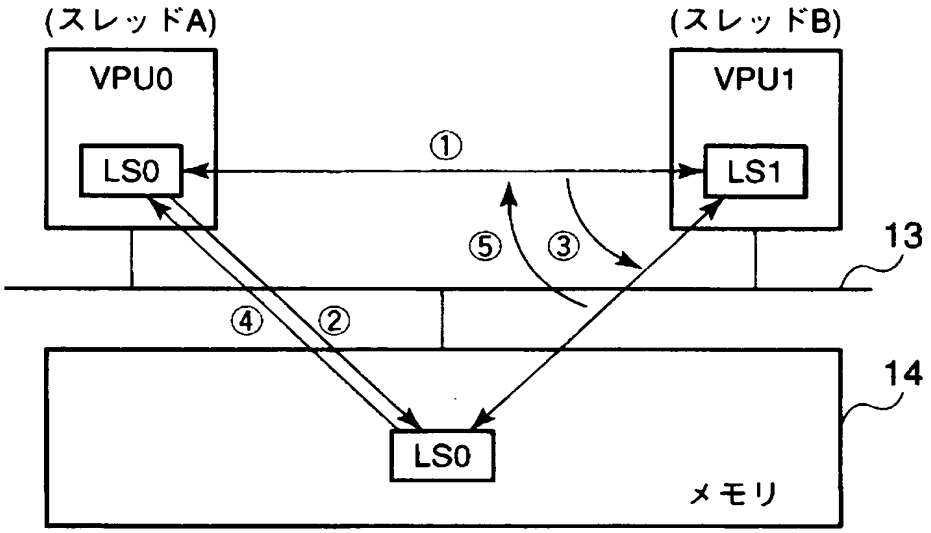
【図４２】


物理VPU0のローカルストレージ
物理VPU1のローカルストレージ
物理VPU2のローカルストレージ
物理VPU3のローカルストレージ
RA空間
VA空間
EA空間
セグメント
スレッドA
スレッドB

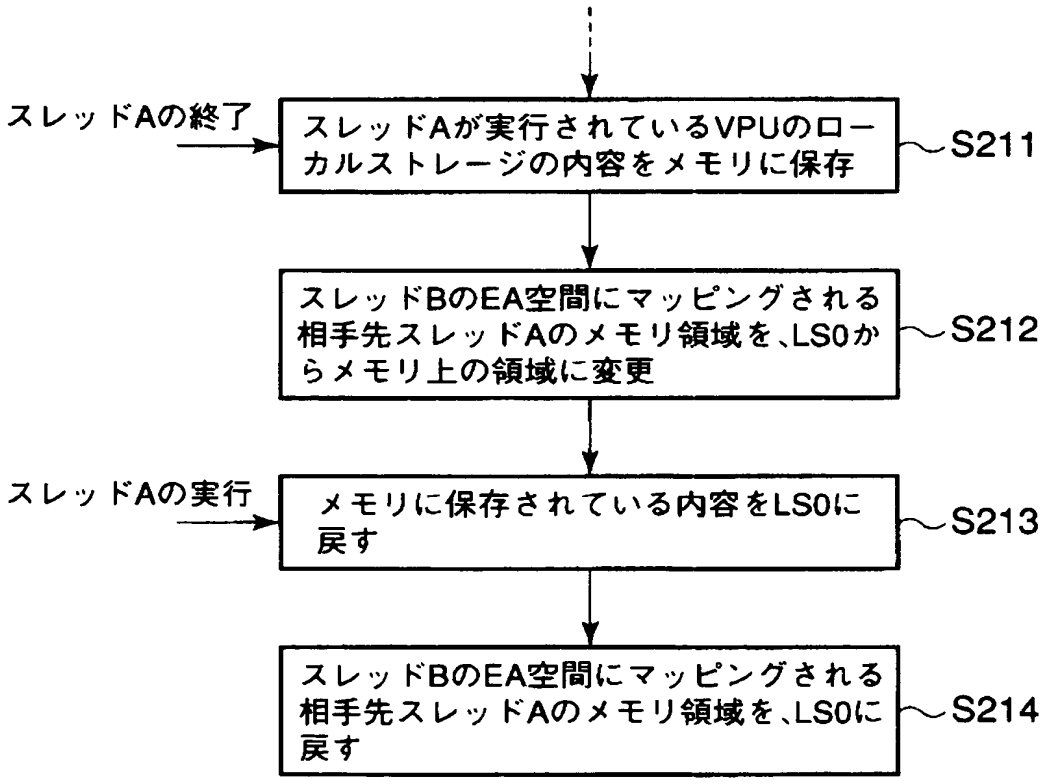
【図４３】


アドレス管理処理
スレッド毎に、そのスレッドのEA空間の固定アドレスに相手スレッドを実行しているVPUのローカルストレージをマッピング
S201
S202
相手スレッドが実行されるVPUが変更されたか？
NO
YES
S203
EA空間の固定アドレスにマッピングされるローカルストレージを、相手スレッドが実行されるVPUに合せて変更

【図４４】


(スレッドA)
VPU0
LS0
(スレッドB)
VPU1
LS1
①
②
③
④
⑤
13
14
LS0
メモリ

【図４５】


スレッドAの終了
スレッドAが実行されているVPUのローカルストレージの内容をメモリに保存
S211
スレッドBのEA空間にマッピングされる相手先スレッドAのメモリ領域を、LS0からメモリ上の領域に変更
S212
スレッドAの実行
メモリに保存されている内容をLS0に戻す
S213
スレッドBのEA空間にマッピングされる相手先スレッドAのメモリ領域を、LS0に戻す
S214

【書類名】　　要約書

【要約】

【課題】複数のプロセッサを用いて複数のスレッドを効率よく並列に実行可能なシステムを実現する。

【解決手段】情報処理システムにおいては、複数のプロセッサＬＳ０，ＬＳ１，ＬＳ２にローカルメモリＬＳ０，ＬＳ１，ＬＳ２がそれぞれ設けられているので、各スレッドは共有メモリをアクセスせずとも、プロセッサ内のローカルメモリをアクセスするだけでプログラムを実行することができる。また、相互作用を行う相手のスレッドが実行されるプロセッサに応じて、実効アドレスＥＡ空間にマッピングされる、相手のスレッドに対応するプロセッサのローカルメモリが自動的に変更されるので、各スレッドは相手のスレッドがディスパッチされるプロセッサを意識することなく、相手のスレッドとの相互作用を効率よく行うことが出来る。よって、複数のスレッドを効率よく並列に実行することが可能となる。

【選択図】　　図３６

特願２００３－１８５４１６

出　願　人　履　歴　情　報

識別番号　　［０００００３０７８］

１．変更年月日　　２００１年　７月　２日
［変更理由］　　住所変更
住　所　　東京都港区芝浦一丁目１番１号
氏　名　　株式会社東芝